FastBERT: a Self-distilling BERT with Adaptive Inference Time

주세준

- BERT, GPT, XLNet

→ good accuracy, but greater costs in computation and slower speed in inference

- NLP datasets have samples of different difficulty

→ FastBERT

- Sample-wise adaptive mechanism

- Self-distillation mechanism

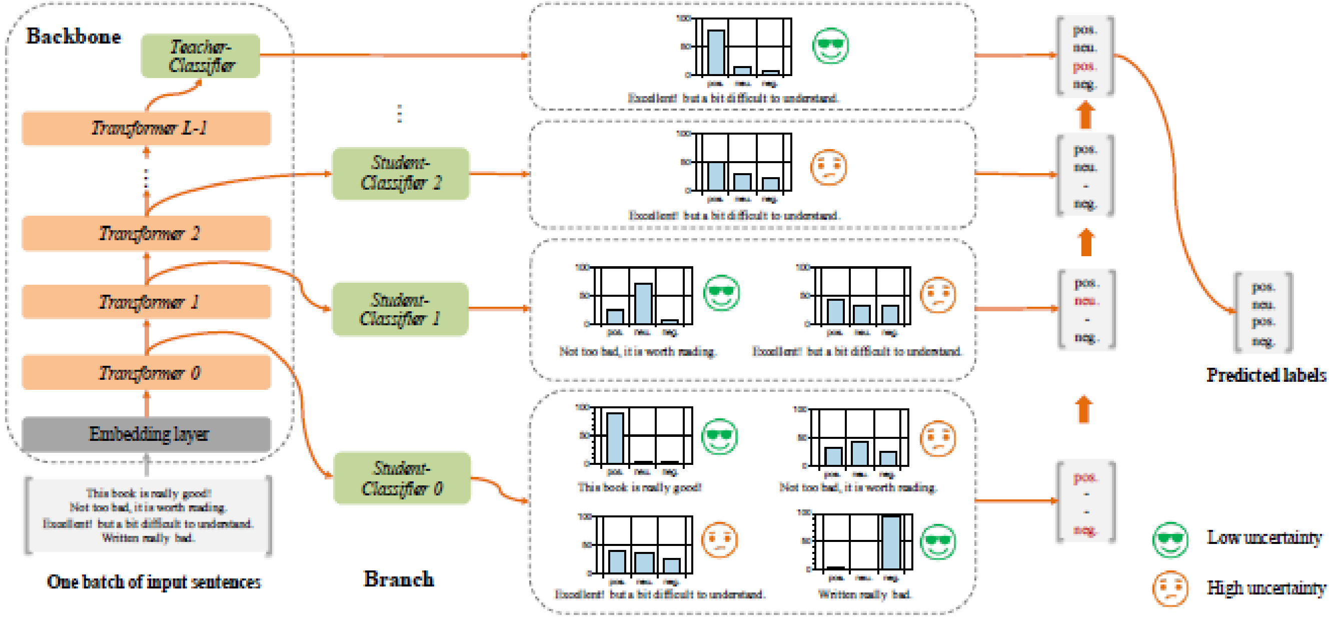


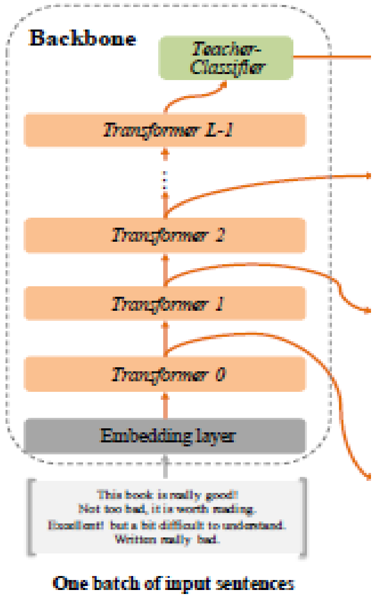
Figure 2: The inference process of FastBERT, where the number of executed layers with each sample varies based on its complexity. This illustrates a sample-wise adaptive mechanism. Taking a batch of inputs ($batch\_size = 4$) as an example, the *Transformer0* and *Student-classifier0* inferred their labels as probability distributions and calculate the individual uncertainty. Cases with low uncertainty are immediately removed from the batch, while those with higher uncertainty are sent to the next layer for further inference.

Model Architecture

- Backbone
  - 12-layer Transformer encoder + teacher classifier
- Branch
  - Student-classifiers (appended to each Transformer output)

Backbone

- Embedding layer
- Encoder (Transformer blocks)
- Teacher classifier

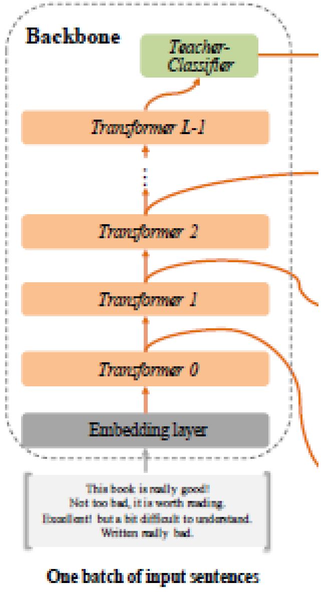


Backbone

- Embedding + Transformer (BERT)
- Teacher Classifier

  Extracts in-domain features for task

  Fully connected layer (768 → 128)
  Self-attention layer
  Fully connected layer + softmax
  N-class indicator



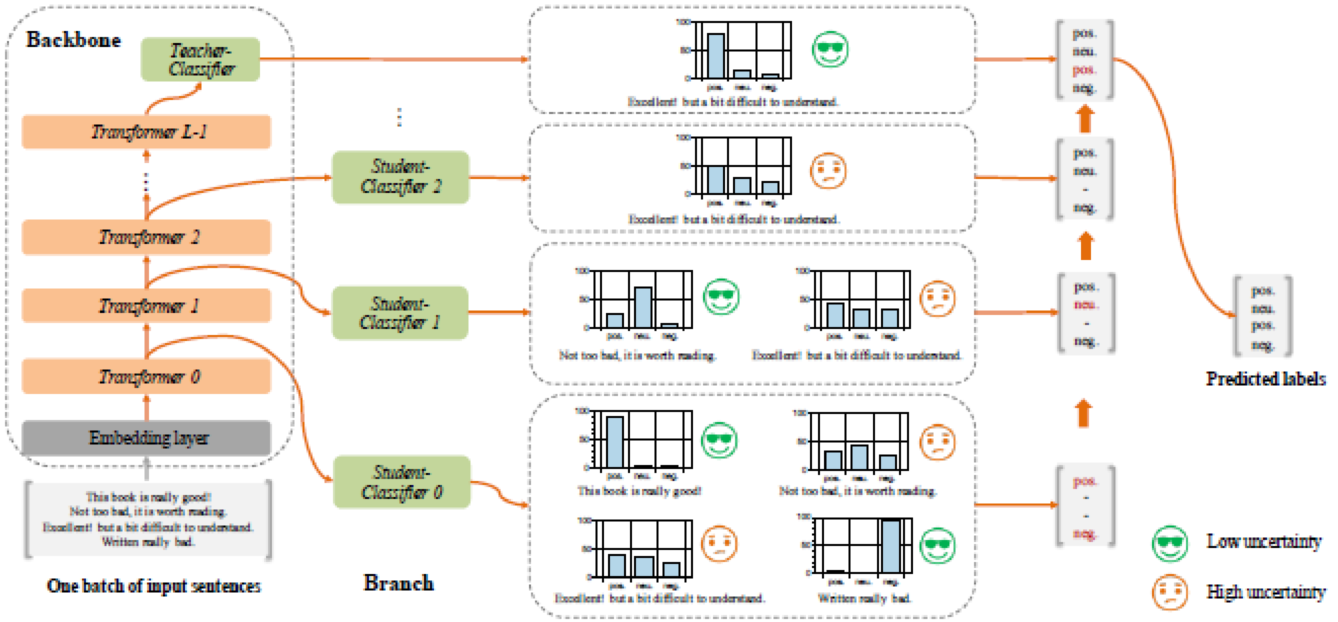
$p_t = Teacher\_Classifier(h_{L-1}).$

$h_i = Transformer\_i(h_{i-1}),$

$e = Embedding(s),$

Branches

$$p_{s_i} = Student\_Classifier\_i(h_i).$$



Model Training (3 major steps)

- Backbone pre-training
- Backbone fine-tuning
- Self-distillation for student classifier

Self-distillation for branch

- p_S student prediction p_t teacher prediction

- Only use teacher output
  → we are free to use an unlimited number of unlabeled data

$$D_{KL}(p_s, p_t) = \sum_{i=1}^{N} p_s(i) \cdot \log \frac{p_s(i)}{p_t(j)}.$$

$$Loss(p_{s_0}, ..., p_{s_{L-2}}, p_t) = \sum_{i=0}^{L-2} D_{KL}(p_{s_i}, p_t),$$

Adaptive inference

- **LUHA** $$Uncertainty = \frac{\sum_{i=1}^{N} p_s(i) \log p_s(i)}{\log \frac{1}{N}},$$
  - The Lower the Uncertainty, the Higher the Accuracy
- **Speed**
  - The threshold to distinguish high and low uncertainty

- Each layer of FastBERT of the corresponding student classifier will predict the label **Uncertainty**

- Samples with Uncertainty **above speed** will move on to the next layer

LUHA verification

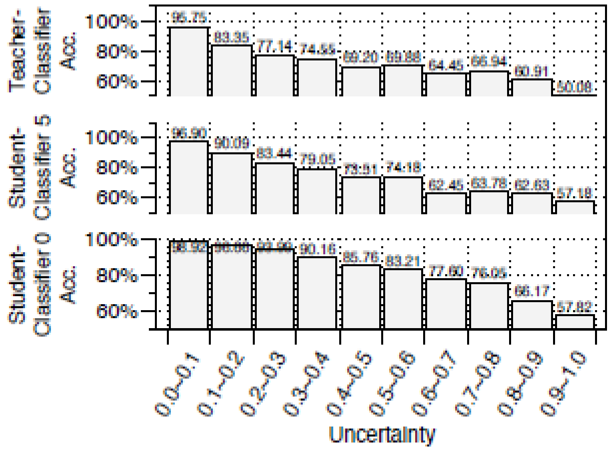


Figure 4: The relation of classifier accuracy and average case uncertainty: Three classifiers at the bottom, in the middle, and on top of the FastBERT were analyzed, and their accuracy within various uncertainty intervals were calculated with the Book Review dataset.

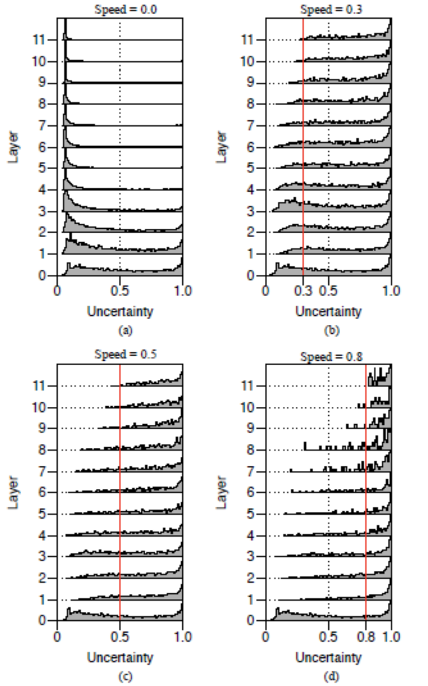


Figure 6: The distribution of *Uncertainty* at different layers of FastBERT in the Book review dataset: **(a)** The *speed* is set to 0.0, which means that all samples will pass through all the twelve layers; **(b)** $\sim$ **(d)**: The *Speed* is set to 0.3, 0.5, and 0.8 respectively, iand only the samples with *Uncertainty* higher than *Speed* will be sent to the next layer.

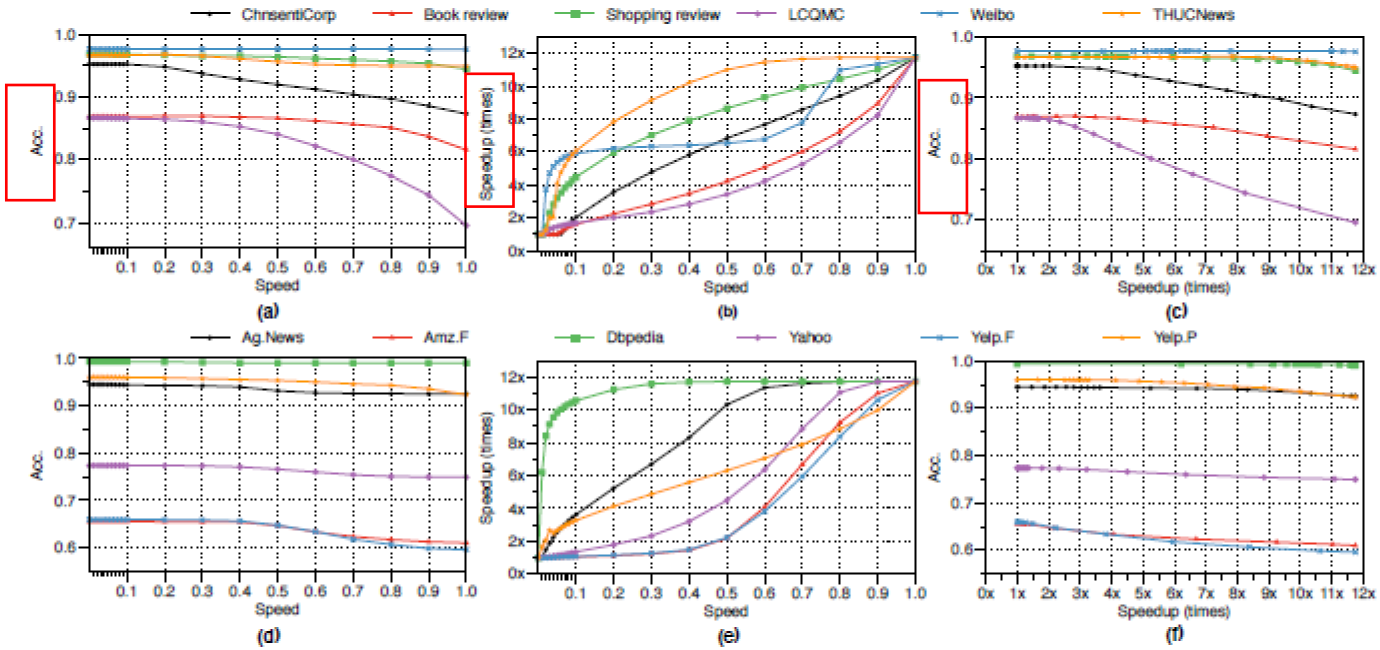


Figure 3: The trade-offs of FastBERT on twelve datasets (six in Chinese and six in English): (**a**) and (**d**) are *Speed-Accuracy* relations, showing changes of *Speed* (the threshold of *Uncertainty*) in dependence of the accuracy; (**b**) and (**e**) are *Speed-Speedup* relations, indicating that the *Speed* manages the adaptibility of FastBERT; (**c**) and (**f**) are the *Speedup-Accuracy* relations, i.e. the trade-off between efficiency and accuracy.

- acc는 fine-tuning 동안 올라가고 self distillation 동안 많이 떨어지는 것을 볼 수 있다.

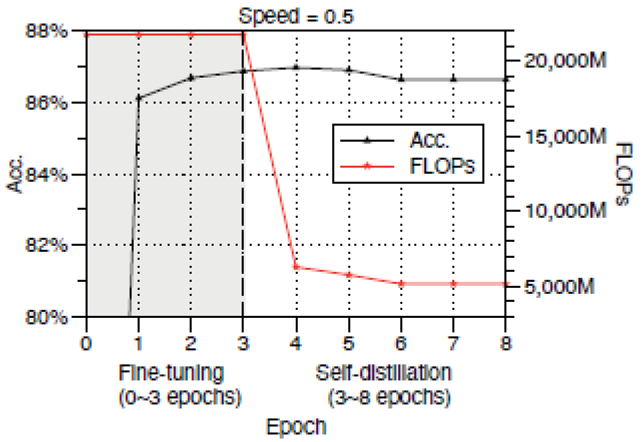


Figure 7: The change in accuracy and FLOPs of Fast-BERT during fine-tuning and self-distillation with the Book review dataset. The accuracy firstly increases at the fine-tuning stage, while the self-distillation reduces the FLOPs by six times with almost no loss in accuracy.

Table 2: Comparison of accuracy (Acc.) and FLOPs (speedup) between FastBERT and Baselines in six Chinese datasets and six English datasets.

Dataset/ Model	ChnSentiCorp		Book review		Shopping review		LCQMC		Weibo		THUCNews	
	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)
BERT	95.25	21785M (1.00x)	86.88	21785M (1.00x)	96.84	21785M (1.00x)	86.68	21785M (1.00x)	97.69	21785M (1.00x)	96.71	21785M (1.00x)
DistilBERT (6 layers)	88.58	10918M (2.00x)	83.31	10918M (2.00x)	95.40	10918M (2.00x)	84.12	10918M (2.00x)	97.69	10918M (2.00x)	95.54	10918M (2.00x)
DistilBERT (3 layers)	87.33	5428M (4.01x)	81.17	5428M (4.01x)	94.84	5428M (4.01x)	84.07	5428M (4.01x)	97.58	5428M (4.01x)	95.14	5428M (4.01x)
DistilBERT (1 layers)	81.33	1858M (11.72x)	77.40	1858M (11.72x)	91.35	1858M (11.72x)	71.34	1858M (11.72x)	96.90	1858M (11.72x)	91.13	1858M (11.72x)
FastBERT (speed=0.1)	95.25	10741M (2.02x)	86.88	13613M (1.60x)	96.79	4885M (4.45x)	86.59	12930M (1.68x)	97.71	3691M (5.90x)	96.71	3595M (6.05x)
FastBERT (speed=0.5)	92.00	3191M (6.82x)	86.64	5170M (4.21x)	96.42	2517M (8.65x)	84.05	6352M (3.42x)	97.72	3341M (6.51x)	95.64	1979M (11.00x)
FastBERT (speed=0.8)	89.75	2315M (9.40x)	85.14	3012M (7.23x)	95.72	2087M (10.04x)	77.45	3310M (6.57x)	97.69	1982M (10.09x)	94.97	1854M (11.74x)

Dataset/ Model	Ag.news		Amz.F		Dbpedia		Yahoo		Yelp.F		Yelp.P	
	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)
BERT	94.47	21785M (1.00x)	65.50	21785M (1.00x)	99.31	21785M (1.00x)	77.36	21785M (1.00x)	65.93	21785M (1.00x)	96.04	21785M (1.00x)
DistilBERT (6 layers)	94.64	10872M (2.00x)	64.05	10872M (2.00x)	99.10	10872M (2.00x)	76.73	10872M (2.00x)	64.25	10872M (2.00x)	95.31	10872M (2.00x)
DistilBERT (3 layers)	93.98	5436M (4.00x)	63.84	5436M (4.00x)	99.05	5436M (4.00x)	76.56	5436M (4.00x)	63.50	5436M (4.00x)	93.23	5436M (4.00x)
DistilBERT (1 layers)	92.88	1816M (12.00x)	59.48	1816M (12.00x)	98.95	1816M (12.00x)	74.93	1816M (12.00x)	58.59	1816M (12.00x)	91.59	1816M (12.00x)
FastBERT (speed=0.1)	94.38	6013M (3.62x)	65.50	21005M (1.03x)	99.28	2060M (10.57x)	77.37	16172M (1.30x)	65.93	20659M (1.05x)	95.99	6668M (3.26x)
FastBERT (speed=0.5)	93.14	2108M (10.33x)	64.64	10047M (2.16x)	99.05	1854M (11.74x)	76.57	4852M (4.48x)	64.73	9827M (2.21x)	95.32	3456M (6.30x)
FastBERT (speed=0.8)	92.53	1858M (11.72x)	61.70	2356M (9.24x)	99.04	1853M (11.75x)	75.05	1965M (11.08x)	60.66	2602M (8.37x)	94.31	2460M (8.85x)

Universal-KD: Attention-based Output-Grounded Intermediate Layer Knowledge Distillation

주세준

Propose

- 기존 loss

$$\mathcal{L} = \alpha\mathcal{L}_{CE} + (1-\alpha)\mathcal{L}_{KD}$$
$$\mathcal{L}_{CE}(x, y; \theta) = CE(y, S(x; \theta))$$
$$\mathcal{L}_{KD}(x, y; \theta) = \mathcal{T}^2\, KL\Big(\sigma(\frac{z_t(x; \phi)}{\mathcal{T}}), \sigma(\frac{z_s(x; \theta)}{\mathcal{T}})\Big)$$

- New loss
    - γL_{univ} $\quad \mathcal{L}_{total} = \alpha\mathcal{L}_{CE} + \beta\mathcal{L}_{KD} + \gamma\mathcal{L}_{\text{Univ.}}$

Universal-KD

Sim 은 similarity function (KL div)

$f_i^t(h_i^t)$ pseudo classifier of ith layer

W_i^t weight of pseudo classifier

λ_{ij} attention weight of j th student layer to i th teacher layer

$F^t(j)$ aggregated predictions

$$\mathrm{Sim}_{\mathrm{Univ.}}(f_i^t(h_i^t), f_j^s(h_j^s)) = KL\Big(\sigma(W_i^t h_i^t), \sigma(W_j^s h_j^s)\Big)$$

$$\lambda_{ij} = \frac{\exp(f_i^t(h_i^t).f_j^s(h_j^s))}{\sum_{i=1}^{N}\exp(f_i^t(h_i^t).f_j^s(h_j^s))}$$

$$\mathcal{F}^t(j) = \sum_{i=1}^{N} \lambda_{ij} f_i^t(h_i^t).$$

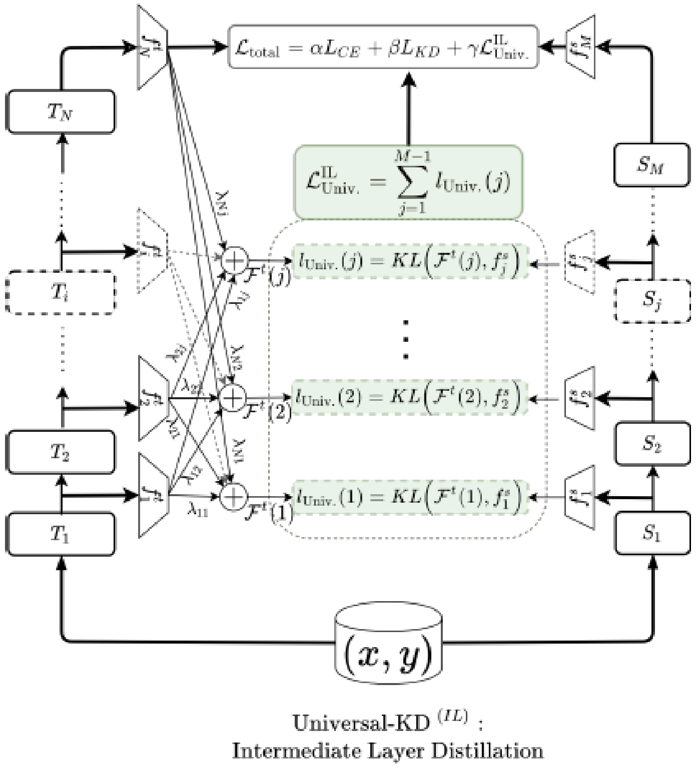


Figure 1: Our Universal-KD in the ILD setting with the pseudo classifiers and attention-based layer projection

$$l_{\mathrm{Univ.}}(j) = \mathrm{Sim}_{\mathrm{Univ.}}(\mathcal{F}^t(j), f_j^s(h_j^s)) = KL\Big(\mathcal{F}^t(j), f_j^s(h_j^s)\Big)$$

Various implementation

- ILD (Intermediate Layer Distillation) M-1 intermediate layers

$$\mathcal{L}^{\text{IL}}_{\text{Univ.}} = \sum_{j=1}^{M-1} l_{\text{Univ.}}(j).$$

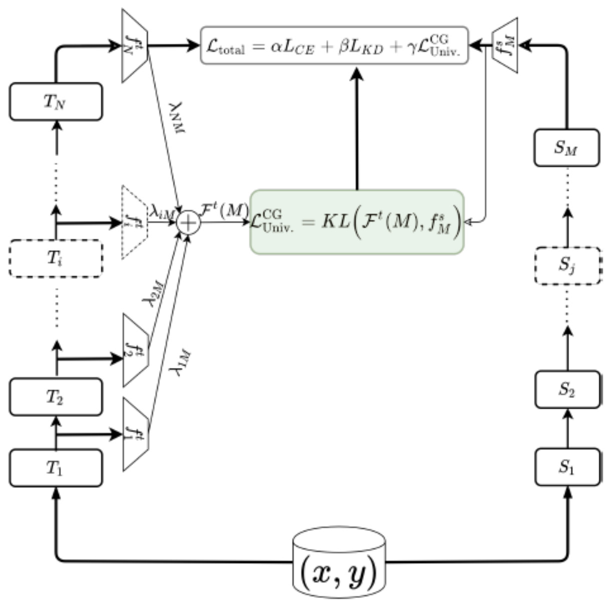
- CG (Capacity Gap)

$$\mathcal{L}^{\text{CG}}_{\text{Univ.}} = l_{\text{Univ.}}(M)$$

- CA (Cross Architecture) same with ILD

$$\mathcal{L}^{\text{CA}}_{\text{Univ.}} = \sum_{j=1}^{M-1} l_{\text{Univ.}}(j)$$

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{CE} + \beta\mathcal{L}_{KD} + \gamma\mathcal{L}_{\text{Univ.}}$$



Universal-KD $^{(CG)}$:
Capacity Gap Problem

Figure 2: Universal-KD to solve the capacity gap problem. Each pseudo classifier acts as a pseudo TA to fill the capacity gap between the two networks.

ILD

- MHKD and ALP, Universal-KD(IL) gives 0.7% and 0.5% performance improvement

Model	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	*Avg*
BERT-base	57.3	83.4	86.8	91.3	91	68.2	92.7	88.9	82.4
BERT-4	31	76.8	77.7	85.1	89	61.7	88.2	87.3	74.6
Vanilla KD	29.2	79.3	79.4	86.8	90.3	65.3	90.4	87.5	76
PKD†	32.1	79.3	80.2	86.6	90.2	65.7	90.1	87.3	76.4
MHKD⋆	32.8	79.4	80.6	86.8	90.1	66.4	90.5	87.5	76.8
ALP†	33.1	**79.6**	80.7	87	90.5	67.2	90.4	87.6	77
Universal-KD$^{(IL)}$	**34.2**	**79.6**	**81**	**87.1**	**90.7**	**67.9**	**90.6**	**87.9**	77.4

Table 2: 4-layer BERT-base student results on GLUE dev set. † denotes the results are taken from (Passban et al., 2020). ⋆ denotes we reproduce this baseline since it is original proposed for CV tasks.

CG

- It is worth noting that our Universal-KD can outperform Annealing
- KD by 0.2% without any temperature adjustment.

Model	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	*Avg*
RoBERTa-large	68.1	90.2	89	94.6	91.5	86.3	96.4	92.3	88.6
DistilRoBERTa	59.3	84	84.3	90.8	90.9	67.9	92.5	88.5	82.3
Vanilla KD	61	84.2	86.5	91.4	91.7	71.1	92.54	88.9	83.4
TAKD†	61.2	83.9	85.8	91.3	**91.7**	71.8	92.5	89	83.4
DIH*	61.5	84.5	86.8	91.5	91.1	72.6	92	88.7	83.6
Annealing KD†	61.7	**85.3**	87.3	**91.6**	91.5	73.6	93.1	89	84.2
Universal-KD(CG)	**63**	83.9	**87.8**	91.6	**91.7**	**74**	**93.5**	**89.9**	**84.4**

Table 4: 6-layer DistilRoBERTa student results on GLUE dev set. † denotes the results are taken from (Jafari et al., 2021). ⋆ denotes we reproduce this baseline since it is original proposed for CV tasks.

CA

- Universal-KD significantly improves the performances of Vanilla-KD by
- 0.8% and 1.4% for Bi-LSTM and Gated-CNN

	Model	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	*Avg*
Teacher	BERT-base	57.3	83.4	86.8	91.3	91.0	68.2	92.7	88.8	82.4
BiLSTM	w/o KD	22	68.6	**70.8**	65.1	84.4	55.6	86.5	18.5	58.9
	Vanilla KD	24.5	69.1	69.4	64.5	84.8	56.3	86.1	20.2	59.4
	Universal-KD$^{(CA)}$	**25.7**	**69.3**	70.6	**65.2**	**85**	**56.5**	**86.9**	**22.1**	**60.2**
Gated-CNN	w/o KD	21.3	58.4	70.6	61.9	81.7	53.4	88	21.8	57.1
	Vanilla KD	20.6	58.7	69.6	62.2	82	54.1	88.8	21.6	57.2
	Universal-KD$^{(CA)}$	**24.5**	**59.1**	**70.8**	**62.4**	**82.2**	**56**	**89**	**24.9**	**58.6**

Table 5: Performances of Bi-LSTM and Gated-CNN students on GLUE dev sets when BERT-base is used as teacher. For each student architecture, we report the scores of models without KD, with vanilla KD, and with Universal-KD.