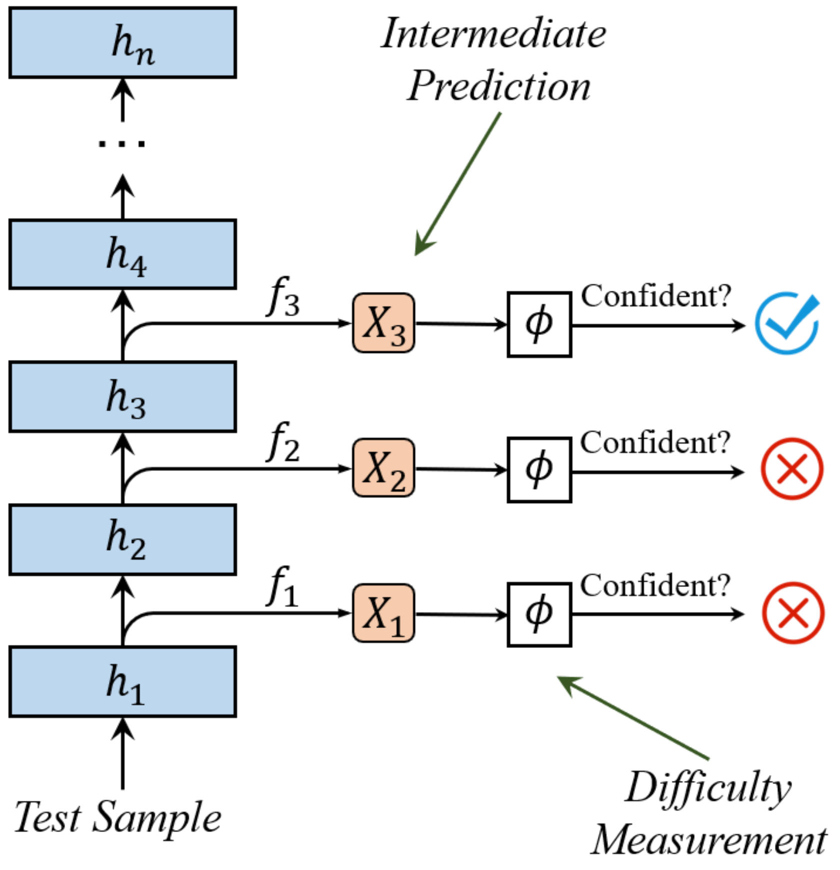
A Simple Hash-Based Early Exiting Approach For Language Understanding and Generation

Tianxiang Sun, Xiangyang Liu, Wei Zhu, Zhichao Geng, Lingling Wu,
Yilong He, Yuan Ni, Guotong Xie, Xuanjing Huang, Xipeng Qiu
Fudan University & Ping An Healthcare Technology

Findings of ACL 2022

Background

- **Early Exiting**
  - Allows instances to exit at early layers according to the **estimation of difficulty**



Background

- **Early Exiting**
  - Allows instances to exit at early layers according to the **estimation of difficulty**
- **How to Estimate Instance Difficulty?**
  - Heuristic metrics, e.g. entropy, maximum softmax score···
    - **Examples**: DeeBERT, FastBERT, PABEE, etc.
    - **Problem**: Suffers from generalization and thresholding tuning.
  - Learning to predict instance difficulty
    - **Example**: Learn-to-exit (Xin et al., EACL 2021)
    - **Problem**: **Can instance difficulty really be learned? (Our initial motivation)**

Can Instance Difficulty Be Learned?

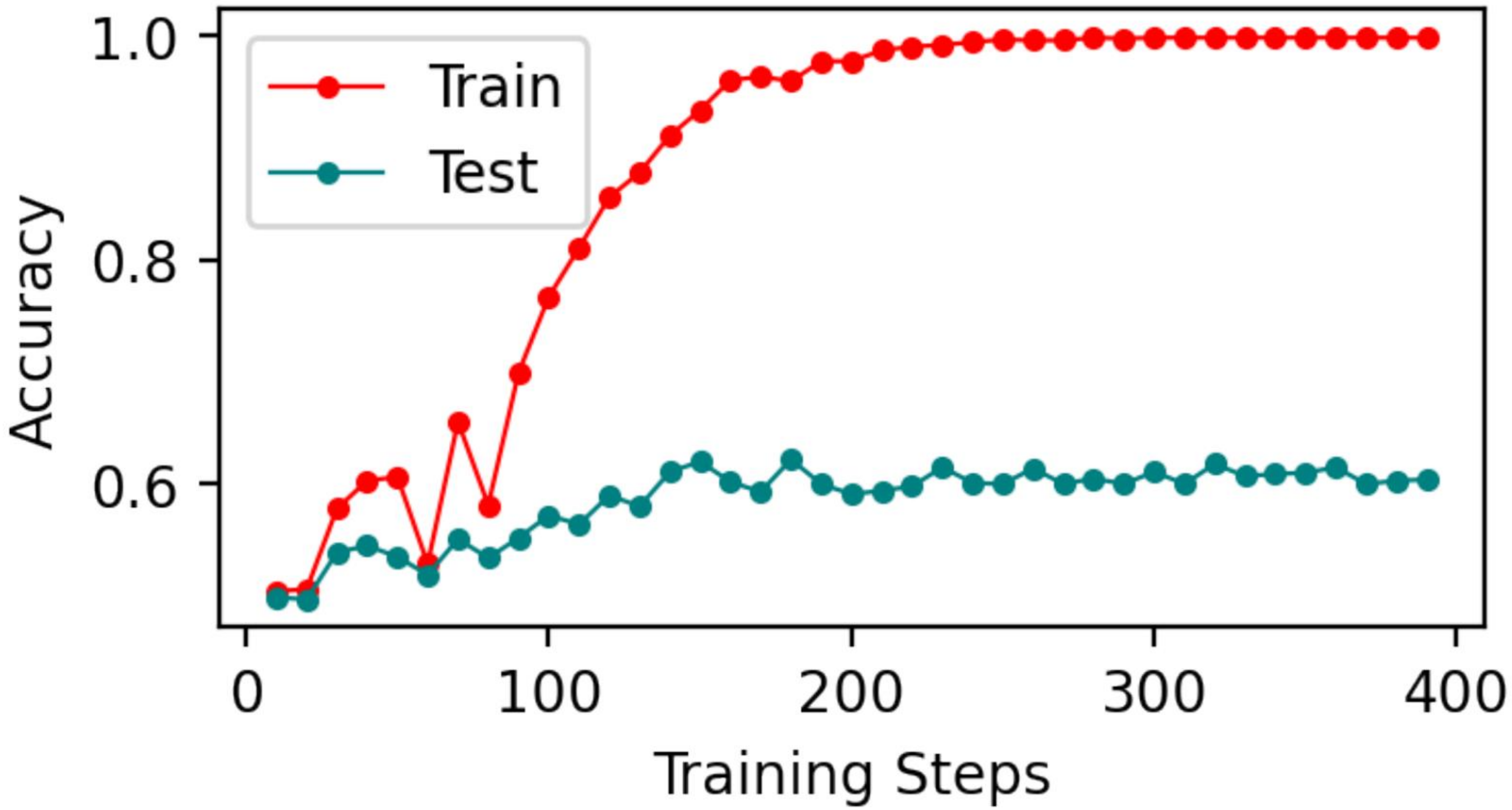
- **How Can We Say Instance Difficulty Can/Cannot Be Learned?**
  - **Step 0:** Define instance difficulty.
  - **Step 1**: We need some "difficulty" datasets.
  - **Step 2**: We use the training set to train a neural model and see if it can generalize to the test set.
- **Step 0 - Two Kinds of Difficulty**
  - **Human-defined difficulty**
    - Measures how difficult for human to judge its label.
  - **Model-defined difficulty**
    - Measures how difficult for a well-trained model to predict its label.

Human-defined Difficulty

- **Step 1: Data Construction**
  - In SNLI, the labels are determined by the majority of the crowd-sourced annotators. If there is no majority for an instance, its label would be “unknown".
  - We collect 1,119 “unknown” instances as **difficulty instances**, and randomly sample 1,119 **simple instances** from the SNLI training set.
  - Now, we have 2,238 instances with two labels (simple or difficult), and randomly sample 1,238 instances with balanced labels as **training set** and use the rest 1k instances as the **test set**.
- **Step 2: Training Models to Predict Instance Difficulty**
  - We train a `BERT-base-uncased` with a linear classifier on the top.

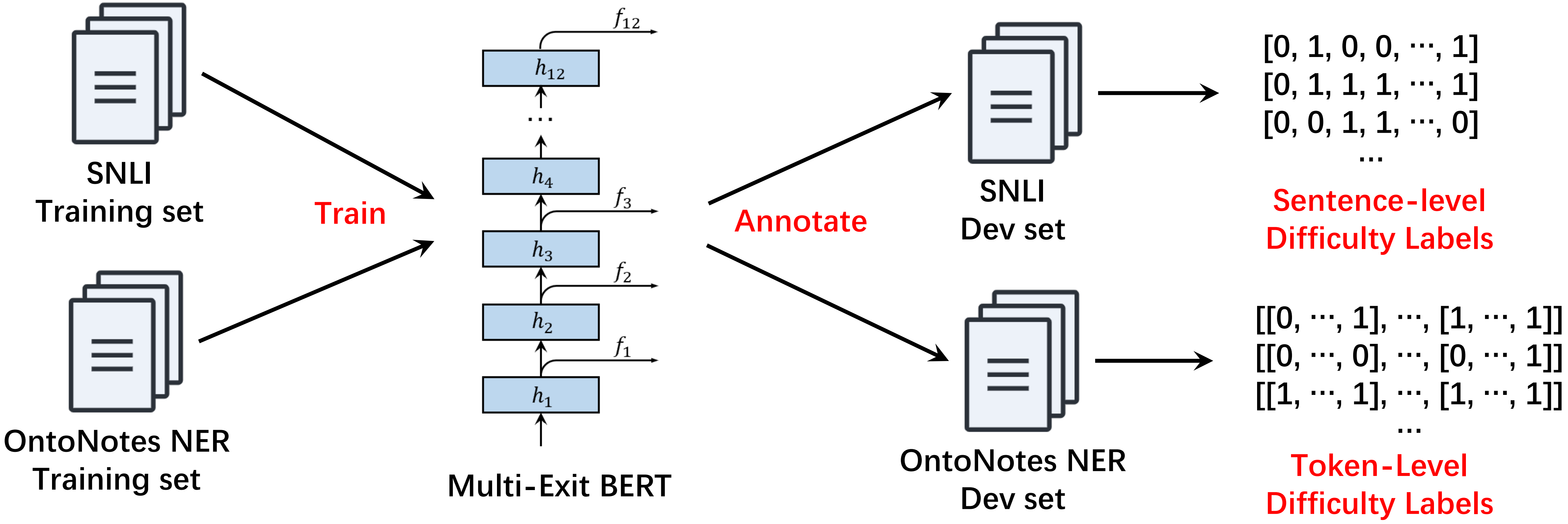
Human-defined Difficulty

- **Can We Learn to Predict Human-defined Difficulty?**
- **Almost no!**



Model-defined Difficulty

- **Step 1: Data Construction**
  - **Idea**: An instance can be defined as a difficult one if it can not be correctly predicted by a well-trained model.

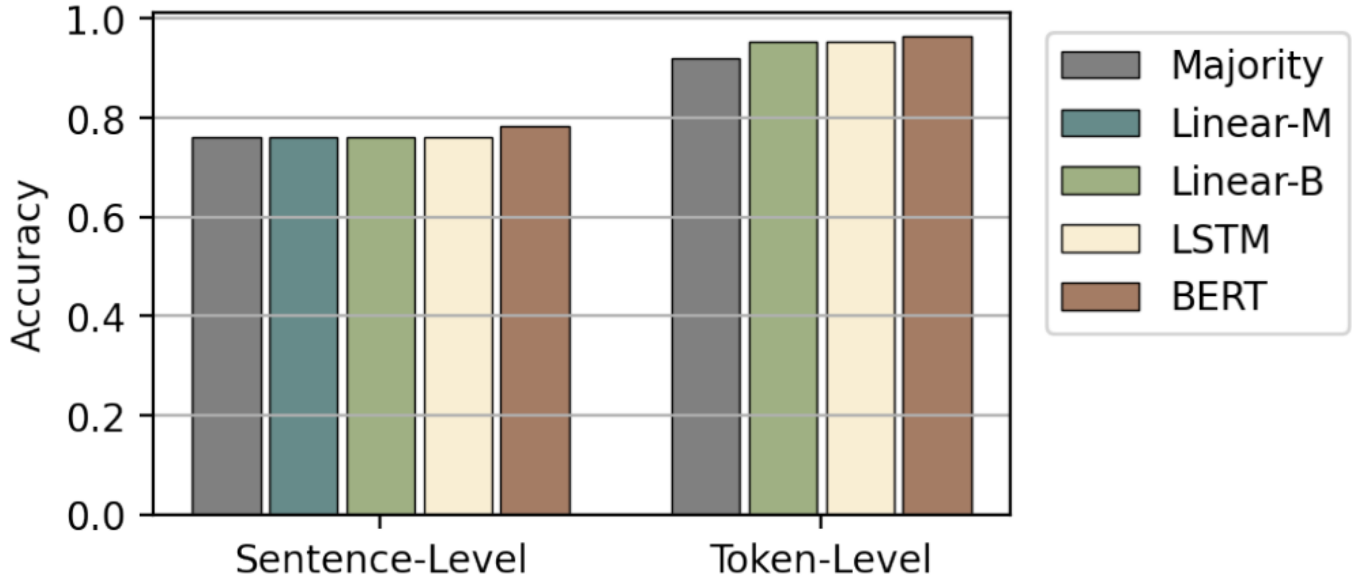


Model-defined Difficulty

- **Step 1: Data Construction**
  - **Idea**: An instance can be defined as a difficult one if it can not be correctly predicted by a well-trained model.
- **Step 2: Training Models to Predict Instance Difficulty**
  - **Majority**: Always predicts the majority class for each label.
  - **Linear-M**: A multi-classification linear layer that takes as input the average pooled word embeddings and outputs the exiting layer (1~12)
  - **Linear-B**: A binary classification linear layer that takes as input the hidden states at each BERT layer and outputs the difficulty label at this layer (0/1)
  - **LSTM**: Takes as input the instance and outputs the exiting layer (1~12)
  - **BERT**: Takes as input the instance and outputs the exiting layer (1~12)

Model-defined Difficulty

- **Can We Learn to Predict Model-defined Difficulty?**
- **Almost no!**



Model	Precision	Recall	F1 Score
Sentence-Level Difficulty			
Majority	60.5	36.7	45.7
Linear-M	54.8	42.1	47.6
Linear-B	52.9	45.3	48.8
BiLSTM	54.5	45.2	49.4
BERT	61.1	49.9	54.9
Token-Level Difficulty			
Majority*	-	-	-
Linear-B	56.6	38.7	46.0
BiLSTM	46.8	39.9	43.0
BERT	65.6	44.6	53.1

HashEE: Hash Early Exiting

- **What Is Unnecessary and What Works?**
  - **On the one hand,** our experiments show that instance difficulty is hard to be predicted.
  - **On the other hand,** learn-to-exit methods have achieved competitive results.
  - **There must be something works and it has nothing to do with estimating instance difficulty.**
  - Let's take a closer look at the learn-to-exit module

$$P(y|x) = \sum_{\substack{d \in D \\ \uparrow \\ \text{Difficulty (implies Architecture)}}} P(y|x,d)\underbrace{P(d|x)}_{\substack{\uparrow \\ \text{Learn-to-exit module}}}$$

HashEE: Hash Early Exiting

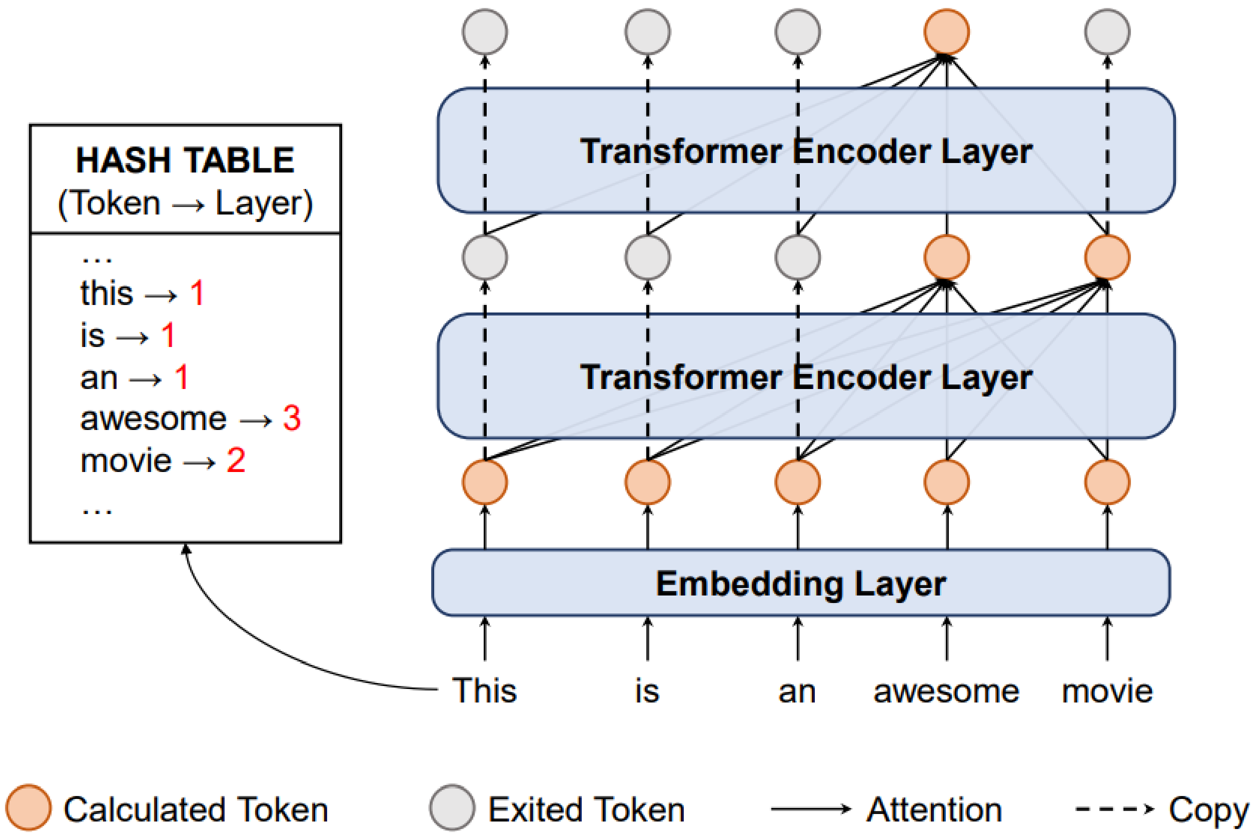
- **What Is Unnecessary and What Works?**
  - **On the one hand,** our experiments show that instance difficulty is hard to be predicted.
  - **On the other hand,** learn-to-exit methods have achieved competitive results.
  - **There must be something works and it has nothing to do with estimating instance difficulty.**
  - Let's take a closer look at the learn-to-exit module
  - **Consistency hypothesis:** *If a training instance x_i is predicted to exit at layer l, then an inference instance x_j that is similar with x_i should exit at layer l, too.*

↓

Can we just replace the neural learn-to-exit module with a simple hash function?

HashEE: Hash Early Exiting

- **HashEE**
  - **Assign tokens to fixed exiting layers using a hash function.**
  - **Considered hash functions:**
    - Random Hash
    - Frequency Hash
    - MI Hash
    - Clustered Hash
  - **Can be used for both NLU and NLG**



Experiments on NLU

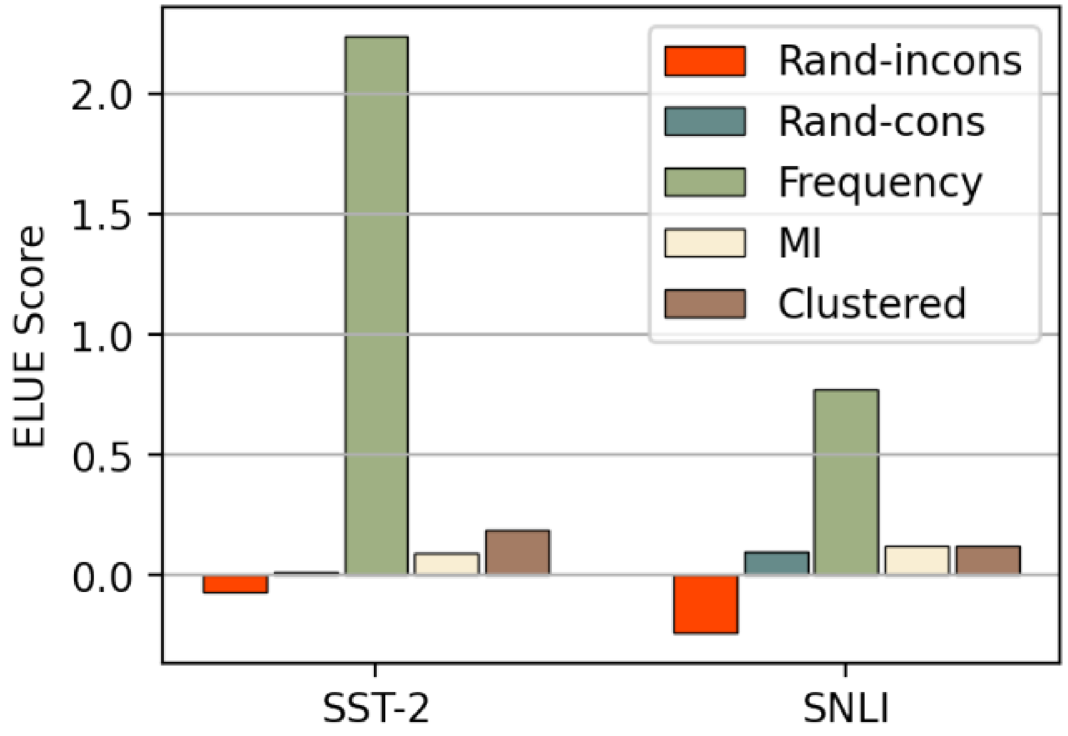
- **State-of-the-art on ELUE (A Benchmark for Efficient NLP)**

Models	**SST-2** (8.5k)	**IMDb** (20.0k)	**SNLI** (549.4k)	**SciTail** (23.6k)	**MRPC** (3.7k)	**STS-B** (5.7k)	**ELUE Score**
Pre-Trained Language Models							
BERT-3L	79.3 (4.0×)	88.4 (4.0×)	87.1 (4.0×)	84.3 (4.0×)	76.0 (4.0×)	75.8 (4.0×)	-3.70
ALBERT-3L	82.4 (3.6×)	90.7 (3.9×)	87.8 (3.7×)	87.5 (3.9×)	80.0 (3.6×)	79.1 (3.9×)	-1.59
RoBERTa-3L	81.8 (4.1×)	90.7 (4.2×)	88.0 (3.8×)	84.9 (3.9×)	75.6 (3.9×)	67.5 (3.9×)	-2.17
ElasticBERT-3L	84.1 (4.0×)	91.8 (4.0×)	89.3 (4.0×)	91.9 (4.0×)	83.1 (4.0×)	83.5 (4.0×)	0.00
Static Models							
DistilBERT	84.8 (2.0×)	92.0 (2.0×)	89.2 (2.0×)	89.7 (2.0×)	83.8 (2.0×)	81.7 (2.0×)	-2.55
TinyBERT	<u>85.3</u> (2.0×)	89.0 (2.0×)	89.3 (2.0×)	90.0 (2.0×)	**84.7** (2.0×)	**85.0** (2.0×)	-2.20
HeadPrune	84.8 (1.3×)	84.7 (1.5×)	87.8 (1.5×)	88.3 (1.5×)	77.8 (1.5×)	74.8 (1.5×)	-6.85
BERT-of-Theseus	84.4 (2.0×)	90.7 (2.0×)	<u>89.4</u> (2.0×)	<u>92.1</u> (2.0×)	82.4 (2.0×)	**85.0** (2.0×)	-2.55
Dynamic Models							
DeeBERT	78.9 (3.4×)	79.5 (4.1×)	48.1 (3.6×)	71.9 (3.4×)	79.1 (3.5×)	-	-
FastBERT	82.7 (3.7×)	**92.5** (3.5×)	88.8 (3.5×)	89.0 (3.6×)	80.3 (4.2×)	-	-
PABEE	83.1 (2.9×)	91.6 (3.4×)	88.7 (3.1×)	90.7 (3.3×)	75.2 (3.5×)	80.1 (3.2×)	-1.31
CascadeBERT	82.4 (3.8×)	91.8 (3.7×)	89.0 (3.6×)	91.7 (3.8×)	78.8 (3.8×)	-	-
BERxiT w/ BERT	71.8 (2.2×)	85.0 (2.8×)	88.4 (3.6×)	80.3 (3.4×)	74.9 (4.0×)	57.8 (4.0×)	-6.12
BERxiT w/ ElasticBERT	72.6 (4.4×)	91.2 (4.0×)	84.7 (3.9×)	91.0 (4.0×)	78.6 (4.3×)	81.5 (4.0×)	-3.90
Ours							
HashEE	**85.5** (**4.8**×)	<u>92.4</u> (**6.2**×)	**89.6** (**4.4**×)	**92.3** (**5.1**×)	<u>84.0</u> (**4.8**×)	84.3 (**4.6**×)	**1.20**

Experiments on NLU

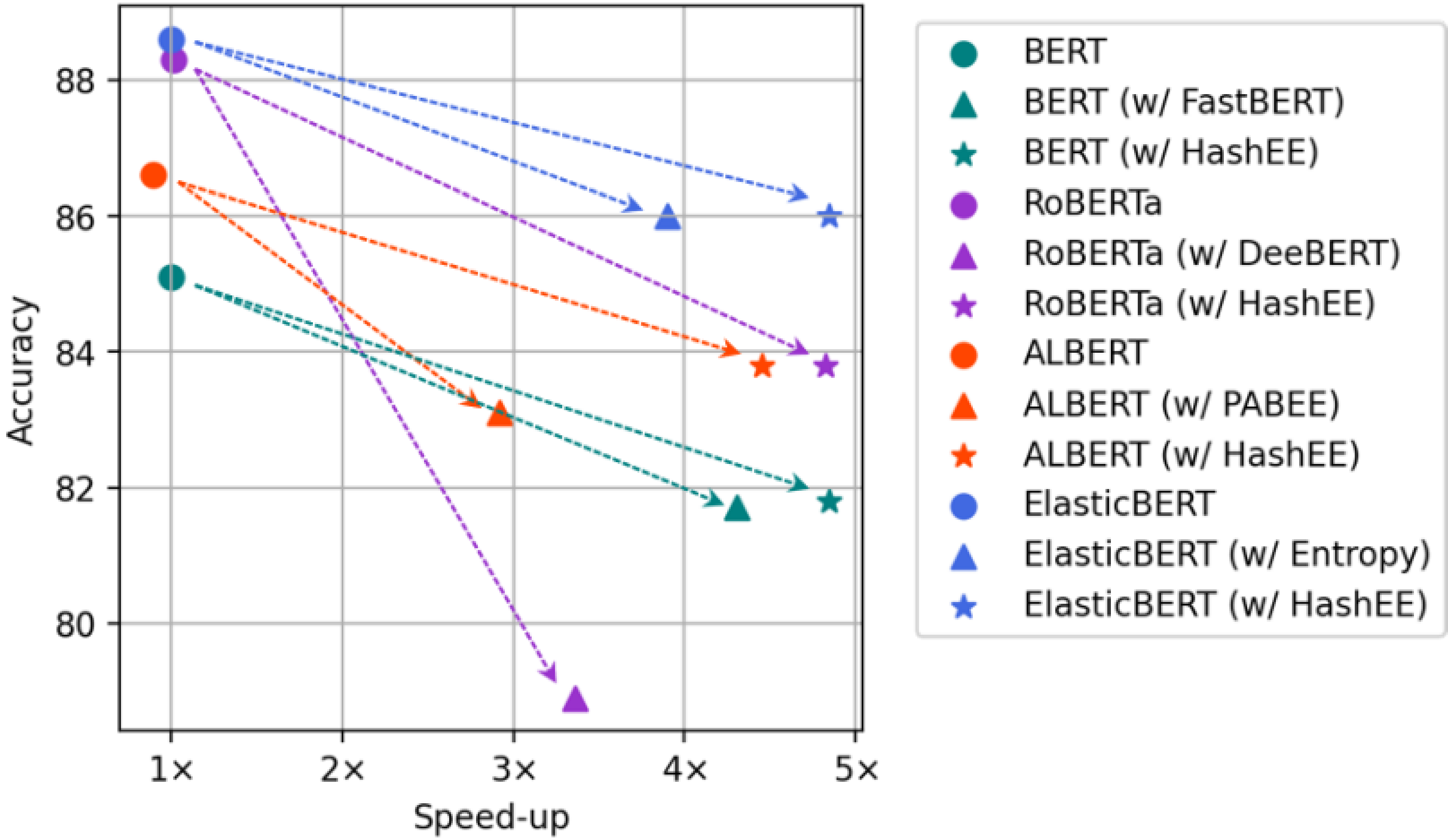
- **Effect of Hash Functions**

Hash Functions	Speed -up	SST-2 (8.5k)	SNLI (549.4k)	MRPC (3.7k)
Backbone: ElasticBERT-6L				
Rand-incons	3.0×	85.5 (±0.53)	89.7	85.0 (±0.22)
Rand-cons	3.0×	85.7 (±0.45)	90.1	86.3 (±0.67)
Frequency	4.9×	85.5 (±0.41)	89.6	84.0 (±0.27)
MI	3.3×	85.5 (±0.49)	90.0	86.0 (±0.23)
Clustered	3.0×	85.7 (±0.50)	90.2	86.3 (±0.47)
Backbone: ElasticBERT-12L				
Rand-incons	1.6×	85.7 (±0.38)	89.6	86.6 (±0.45)
Rand-cons	1.5×	86.5 (±0.37)	90.2	87.4 (±0.34)
Frequency	2.8×	85.6 (±0.37)	89.8	84.4 (±0.17)
MI	1.8×	86.6 (±0.17)	90.1	87.2 (±0.66)
Clustered	1.5×	87.0 (±0.54)	90.1	87.3 (±0.48)



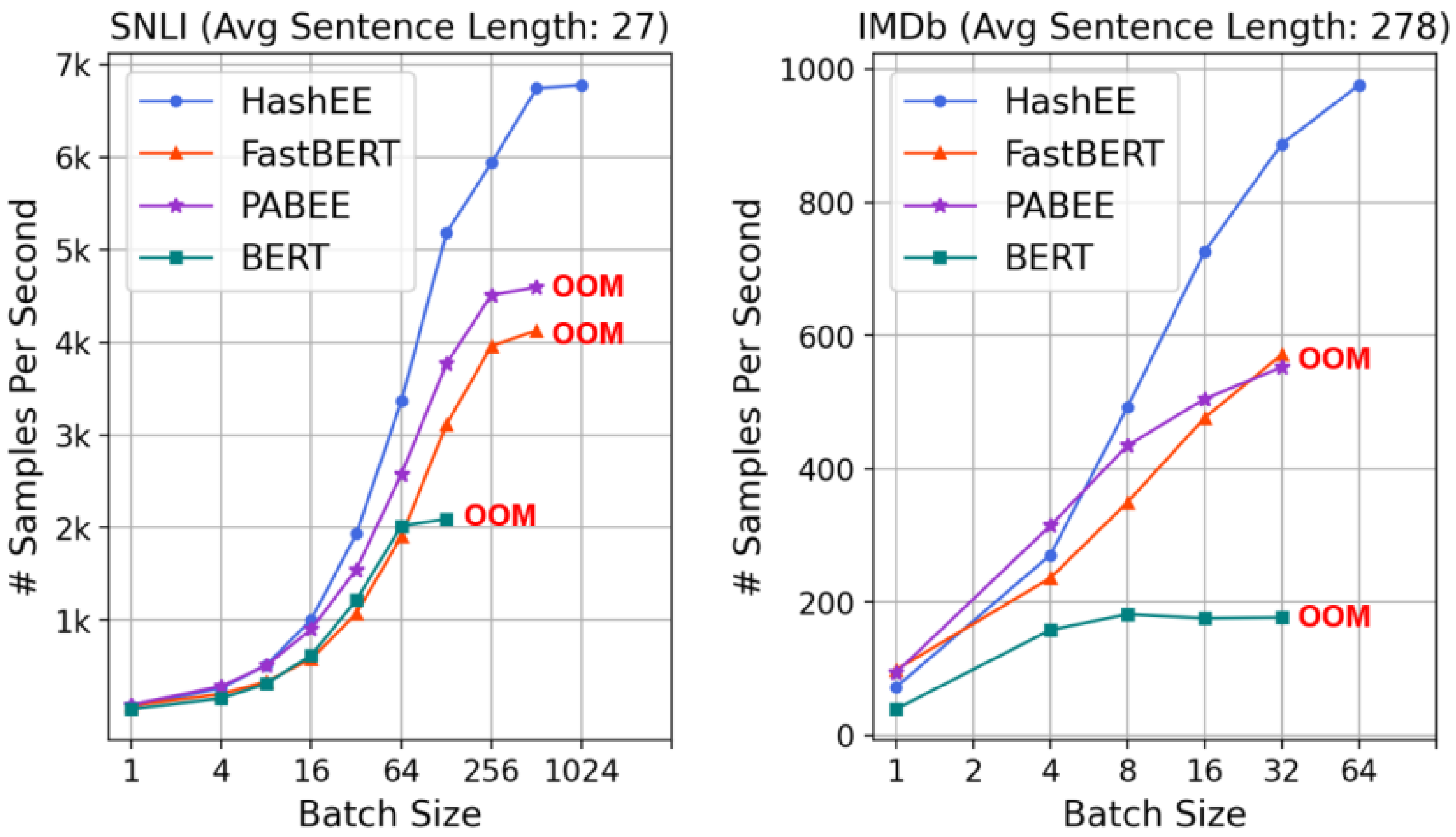
Experiments on NLU

- **Effect of Backbones**



Experiments on NLU

• **Comparison of Actual Inference Time**



Experiments on NLG

- **Results on 4 Summarization Datasets**

Model	Speed-up			English		Chinese	
	Enc.	Dec.	Total	Reddit	CNN/DM	CSL	TTNews
BART	1.0×	1.0×	1.0×	29.71/9.91/23.43	44.16/21.28/40.90	64.49/52.48/61.81	53.84/38.09/49.85
DAT	1.0×	0.5×	0.8×	27.02/**8.89/22.68**	40.30/17.77/37.53	-	-
BART-6L	2.0×	**1.4**×	**1.8**×	26.22/6.82/21.05	40.02/16.60/36.82		-
HashEE w/ BART	**3.3**×	1.0×	**1.8**×	**28.77**/8.52/21.97	**41.04/18.41/37.65**	-	-
CPT	1.0×	1.0×	1.0×	-	-	65.49/53.82/62.96	53.48/37.59/49.82
CPT-6L	2.0×	**1.2**×	1.9×	-	-	52.29/39.35/50.06	50.89/33.75/45.42
HashEE w/ CPT	**2.3**×	1.0×	**2.2**×	-	-	**62.42/49.96/59.15**	**52.67/35.31/46.97**

Thanks!