PYRAMID DYNAMIC INFERENCE: ENCOURAGING FASTER INFERENCE VIA EARLY EXIT BOOSTING

Ershad Banijamali *Pegah Kharazmi* *Sepehr Eghbali*
Jixuan Wang *Clement Chung* *Samridhi Choudhary*

Amazon Alexa AI

ABSTRACT

Transformer-based models demonstrate state of the art results on several natural language understanding tasks. However, their deployment comes at the cost of increased footprint and inference latency, limiting their adoption to real-time applications. Early exit strategies are designed to speed-up the inference by routing out a subset of samples at the earlier layers of the model. Exiting early causes losing model accuracy. In order to optimize the trade-off between model accuracy and latency, we propose Pyramid Dynamic Inference (PDI), a scheme that encourages fast inference via boosting the performance of early exit heads. PDI allows for more confident early inference by injecting stronger classifiers at earlier layers. It also prevents a significant increase in the model footprint by gradually shrinking the classifiers as the semantic capacity of the deeper transformer layers increase. Experiment results show that PDI outperforms the baselines on both accuracy and latency on the GLUE benchmark.

1. INTRODUCTION

Natural Language Understanding (NLU) refers to a collection of tasks to infer semantics from a text source and is a key technology behind commercial voice assistants (VAs) such as Google Home, Alexa, Siri and more. Recent advancements in deep learning and in particular the development of large transformer-based models have led to substantial accuracy increases in a variety of semantic inference tasks [1, 2, 3]. With the rising usage of VAs because of the everyday convenience they deliver[1], having a single central model is only scalable up to a certain point. Furthermore, more users are concerned about how their data is processed. Therefore, data privacy and low response latency become key factors in ensuring a good user experience in addition to accuracy. 'On-device' or 'Edge' processing has become a key technology for enabling these features [4, 5]. However, edge processing faces limited computational, memory and storage resources, which is a bottleneck in deploying complex transformer-based models [6].

[1] https://www.statista.com/statistics/973815/worldwide-digital-voice-assistant-in-use/

Recently, there have been significant efforts to optimize large models such as BERT for on-device use. Several studies focusing on reducing the model footprint have investigated structure pruning [1, 7], progressive module replacement [8], quantization [9, 10, 11], knowledge distillation [4, 12, 6] and matrix factorization approaches [13, 14, 15, 16]. These studies report achieving 35%-87% size reduction. However, a trade-off is made in accuracy with degradation of up to 10% in some approaches [13]. Moreover, owing to the computational load of the remaining sparse structure of the compressed model in standard GPU implementations, the reduction in model parameters does not always translate to significant inference speedup. Other studies have explored techniques to reduce the runtime latency by introducing dynamic inference and early exit strategies. Authors in [17] proposed inserting exit points at different layers of the network and used the entropy of a classification result as a measure of confidence to route the samples. In [18] authors used the same scheme in a Spoken Language Understanding (SLU) setup. A fine-tuning strategy is proposed in [19] that extends early exiting to tasks other than classification. In most existing techniques, an early exit route is chosen only if its confidence is above a certain threshold. Their major drawback is that since early layers have limited semantic capacity, they demonstrate low confidence on complex sample. Therefore, enforcing early exit would result in accuracy drop. FastBERT [20] tries to reduce the accuracy loss by a speed-tunable model that uses a self-distilling strategy to train the parameters of the intermediate exit heads.

In this work, we aim to provide a better platform of trade-off between accuracy and speedup. We present Pyramid Dynamic Inference (PDI), a scheme that encourages fast inference while retaining the model performance by boosting the early exit heads in VC-dimension sense [21]. PDI makes up for the limited capacity of earlier layers of the model to infer complex semantics by injecting stronger classifiers to boost the confidence at these layers. As we progress to deeper transformer layers and the network can extract more complex semantics, PDI gradually shrinks the classifier, thereby preventing the model footprint from significant increase. Compared to FastBERT our model uses a depth-dependent strategy to choose the complexity of the exit heads and employs a hybrid loss to train the parameters of these heads. The main contributions of

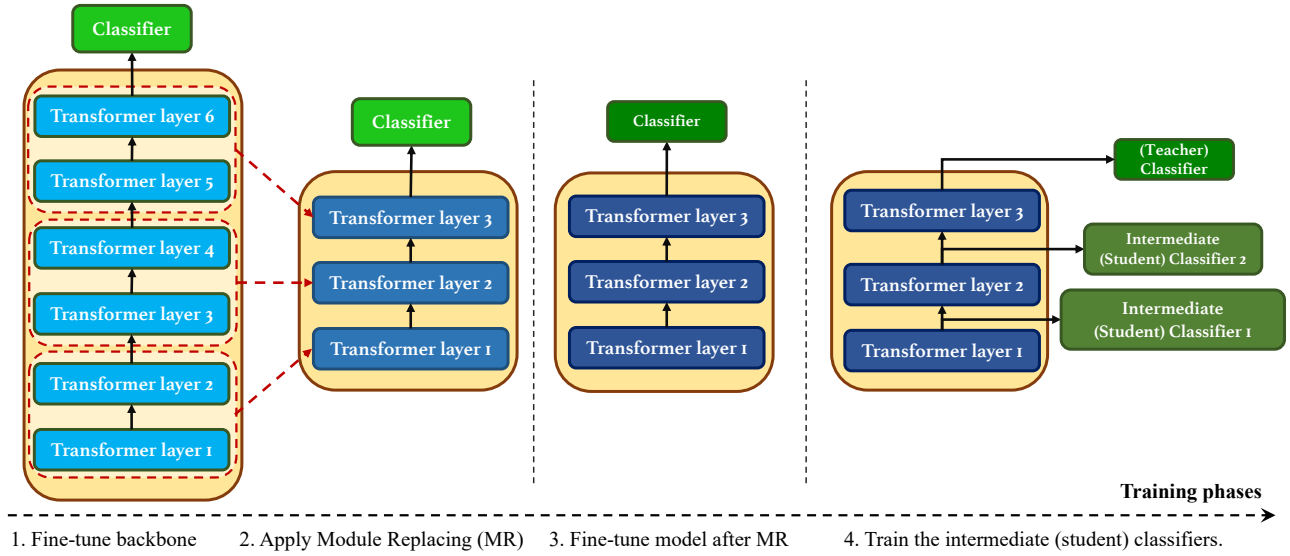

Fig. 1: Training phases of the PDI scheme. Parameter update of each module from one phase to the next is shown through different colors.

this paper are: (1) We propose PDI, a novel early exit scheme, that helps early layers of the model retain confidence in inferring semantically complex samples, thereby encouraging early exit among a larger input space. (2) We optimize the accuracy, latency and footprint trade-off by leveraging PDI in conjunction with model compression, demonstrating a viable transformer model candidate for real-time applications on resource constrained devices.

2. METHODOLOGY

2.1. PDI Architecture

Although our approach is architecture agnostic, we choose DistilBERT as our backbone model, which is an already compressed distilled model, comprised of an embedding layer followed by 6 transformer layers [6]. The output of each layer h_i is defined as:

$$h_i = \ell_i(h_{i-1}). \tag{1}$$

where ℓ_i represents transformer layer i, with h_0 being the output of the embedding layer.

In this study, we focus on sequence classification as the final task of our model. To this purpose, a classifier head is attached to the last transformer layer, which outputs a distribution over the set of possible class labels. To decrease the average inference time of the model, we add intermediate exit classifier heads onto different transformer layers. At inference time, if an intermediate classifier outputs a label with high certainty, further inference is stopped and this output is taken as the final output of the model. We define the uncertainty of the classifier j, denoted by u_j, as the normalized entropy of its output distribution, $\mathbf{p}_j$:

$$u_j = \frac{\sum_{c=1}^{C} \mathbf{p}_j(c) \log \mathbf{p}_j(c)}{\log 1/C}, \tag{2}$$

where C denotes the number of classes. We exit from layer j if $u_j < \tau$, where τ is the certainty threshold. The larger the value of τ, the more chance of exiting from lower layers, i.e. higher inference speed.

Our approach is based on the observation that intermediate classifiers with higher accuracy are more likely to output labels with high certainty [20]. Therefore, we improve the accuracy of intermediate classifiers by using a pyramidal structure for the classifiers: we attach stronger classifiers at the shallower layers of the transformer and decrease the complexity of the classifiers as we go deeper in the transformer layers of the model. The assumption behind this approach is that since earlier layers of the network provide weaker semantic information, a stronger classification head could help balance such weakness. However, adding bigger intermediate classifiers comes at the cost of increasing the total number of model parameters and an increased footprint. To compensate for that, we compress the base model even further using module replacing [8]. To do so, we replace each two transformer layers of DistilBERT with one transformer layer. Our results show that such compression reduces the model size by $\approx 32\%$, while retaining 98.9% of DistilBERT performance. The intermediate classifiers are attached to the layers of the reduced size model.

2.2. Model Training

As shown in figure 1, the training of our proposed PDI scheme consists of four phases. In phase 1, we fine-tune DistilBERT on the downstream task. We then apply module replacing on the fine-tuned DistilBERT in phase 2. Phase 3 performs task-specific fine-tuning on the the model from phase 2. We then use the model from phase 3 as the base model for early exiting, by attaching intermediate classifiers in a pyramidal structure, while other model parameters are frozen.

We use a hybrid loss function to train the intermediate clas-

sifiers: 1) A classification loss with the ground truth labels as the target, and 2) A self-distillation loss that uses the output of the final classifier as the target. In this setup, the final classifier serves as the teacher classifier for training the intermediate (student) classifiers. The hybrid loss is a weighted sum of these two losses:

$$\mathcal{L}_{joint} = \sum_{j=1}^{M-1} \alpha \mathcal{L}_{CL}(\mathbf{p}_j, \mathbf{c}) + (1-\alpha)\mathcal{L}_{KD}(\mathbf{p}_j, \mathbf{p}_M), \quad (3)$$

where M denotes the number of model layers, $\mathbf{c}$ is the one-hot ground truth vector, α is the weight hyperparameter, $\mathcal{L}_{CL}$ is the classification loss, and $\mathcal{L}_{KD}$ is the knowledge (self-) distillation (KD) loss, which is defined by the cross-entropy between the outputs of student and teacher classifiers. We show in section 3.4.2 that using the hybrid loss can improve the performance of the model.

3. EXPERIMENTS

3.1. Datasets

We evaluate our model on the GLUE benchmark [22, 23, 24, 25, 26, 27]. For SST-2, MNLI-m, MNLI-mm, QNLI and RTE we use accuracy as the classification metric. The F1 and accuracy are used for MRPC and QQP. Pearson and Spearman correlation are used for STS-B and CoLA is evaluated based on Matthew's correlation. The results are reported for the development sets of GLUE.

3.2. Baselines

- **DistilBERT:** A 6-layer pretrained DistilBERT with a 2-layer fully-connected classifier fine-tuned on each task, i.e. the model in phase 1 in Fig. 1.
- **DistilBERT with Module Replacing (MR):** A 3-layer compressed DistilBERT using MR and fine-tuned on each task, i.e. the model in phase 3 in Fig. 1.
- **FastBERT on DistilBERT with MR:** A compressed DistilBERT using MR with intermediate classifiers at each layer. All the intermediate classifiers of this model have the same architecture as the final classifier, i.e. a 2-layers fully-connected classifier. Also we only use self-distillation loss to train the intermediate classifiers.

3.3. Experiment Results

To build the pyramid structure, we inject a four-layer fully-connected classifier head at the first layer of the model, a three-layer fully-connected classifier head at the second layer, and a two-layer classifier at the last layer. Table 1 compares the number of parameters of the proposed architecture with the baselines. PDI has $\approx 1.5\%$ more number of parameters compared to FastBERT, due to deeper student classifiers. We set $\alpha = 0.5$ during training, based on the validation set.

Table 1: Comparison of total number of model parameters.

Model	# of param.
DistilBERT	66.9M
DistilBERT with MR	45.6M
FastBERT	46.8M
PDI (our model)	47.5M

Table 2 shows the classification results and relative inference time of different models on the GLUE benchmark. We use DistilBERT with MR as the baseline for the inference time and normalize other models' speedups with respect to that. As expected, DistilBERT is constantly slower (speedup factor of $0.53\times$) compared to the baseline due to having more transformers layers and no early exit heads. We report the results over three different values of certainty thresholds, τ, for FastBERT and PDI. This table demonstrates that across all datasets and values of τ, the proposed PDI scheme outperforms FastBERT on both classification metrics and speedup factors. Since the student classifiers of PDI are more accurate (and more certain about their output as we will see in the next section) compared to that of FastBERT, there is a higher chance that PDI exits in its earlier layers. At τ = 0.4, PDI achieves on average $3.38\times$ and $1.69\times$ speedup compared to the original 6-layer DistilBert and DistilBERT with MR, respectively. While for the same value of τ, there is only 2.5% and 1.5% accuracy degradation compared to the original 6-layer DistilBert and DistilBERT with MR, respectively. Moreover, the classifier heads add negligible complexity to the model compared to other model components, allowing PDI to further improve its speedup compared to other baselines.

3.4. Further Analysis

Here we provide more insight about PDI performance. We study the output of each individual classifier as well as the role of each contributing factor to our model performance. For this investigation we use the QQP dataset, however the results show the same trends for the other datasets.

3.4.1. Layer-wise Analysis

Fig. 2 shows the output of each intermediate classifiers for both PDI and FastBERT. As we can see in the top figures, PDI classifiers are more accurate than FastBERT across all uncertainty intervals. Also, the accuracy of both models degrade as the uncertainty increases. On the other hand, the middle figures show that, for the lower values of uncertainty, the intermediate classifiers of PDI can predict the labels for more samples. For example the first classifier of PDI predicts the label for 53% of the samples with uncertainty less than 0.25, compared to the 43% for FastBERT. The bottom figures show the accuracy and speedup of the models. For larger values of τ the gap between the accuracy of the models enlarges. However, since for a large τ ($\tau > 0.9$) almost all samples exit from

Table 2: Experiment results (median of 5 runs) on the development set of the GLUE benchmark in terms of classification metrics (Cls.) and inference speed-ups.

Model	τ	CoLA (8.5K)		MNLI (393K)		MRPC (3.7K)		QNLI (105K)	
		Cls.	speedup	Cls.	speedup	Cls.	speedup	Cls.	speedup
DistilBERT	-	43.6	0.53×	79.0	0.53×	87.5	0.53×	85.3	0.53×
DistilBERT with MR	-	41.8	1.00×	77.8	1.00×	86.8	1.00×	84.3	1.00×
FastBERT	0.2	41.7	1.26×	76.9	1.24×	84.9	1.42×	84.0	1.66×
	0.4	36.4	1.55×	73.2	1.49×	80.5	1.63×	80.3	1.75×
	0.5	35.3	1.79×	72.6	1.84×	79.5	1.84×	78.9	2.00×
PDI (our model)	0.2	41.8	1.36×	77.2	1.29×	86.2	1.49×	84.0	1.67×
	0.4	40.8	1.83×	76.8	1.45×	85.8	1.71×	83.2	1.82×
	0.5	39.3	2.01×	75.8	1.85×	85.1	2.05×	82.8	2.02×

Model	τ	QQP (364K)		RTE (2.5K)		SST-2 (67K)		STS-B (5.7K)		Avg.
		Cls.	speedup	Cls.	speedup	Cls.	speedup	Cls.	speedup	Cls.
DistilBERT	-	84.9	0.53×	63.4	0.53×	90.7	0.53×	84.7	0.53×	77.38
DistilBERT with MR	-	84.5	1.00×	62.8	1.00×	89.6	1.00×	84.3	1.00×	76.48
FastBERT	0.2	84.2	1.43×	62.2	1.13×	89.0	1.20×	83.7	1.32×	75.82
	0.4	82.7	1.66×	59.5	1.45×	86.8	1.33×	82.0	1.52×	72.72
	0.5	81.3	1.96×	58.3	1.62×	85.9	1.66×	80.9	1.87×	71.65
PDI (our model)	0.2	84.3	1.54×	62.3	1.21×	88.6	1.22×	83.9	1.45×	76.06
	0.4	83.8	2.00×	61.4	1.50×	87.9	1.56×	83.0	1.72×	75.38
	0.5	83.0	2.07×	61.1	1.76×	86.7	1.78×	82.3	1.99×	74.57

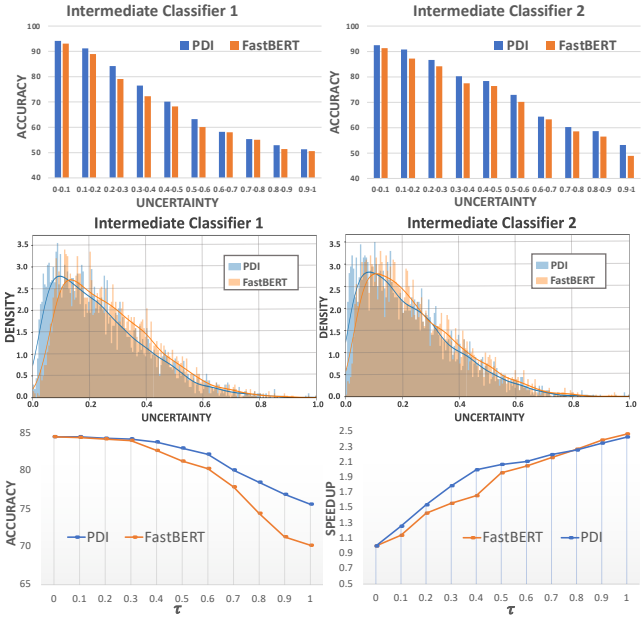

Fig. 2: Top: Accuracy of intermediate classifiers for each uncertainty interval. Middle: Density of samples vs uncertainty for intermediate classifiers. Bottom: Accuracy and Speedup of PDI and FastBERT for each value of τ.

the first layer, vanilla FastBERT has better speedup. This is due to the simpler intermediate classifier in the first layer of FastBERT compared to PDI.

3.4.2. Ablation Study

Here we investigate role of the two distinguishing factors between PDI and FastBERT: the pyramid structure of the intermediate classifiers and the hybrid loss for training them. We consider two scenarios. First, a model for which the intermediate classifiers have pyramid structure but they are trained using only self-distillation loss, i.e. PDI without hybrid loss (PDI w/o HL). Second, a model with vanilla FastBERT structured that is trained using hybrid loss (FastBERT with HL).

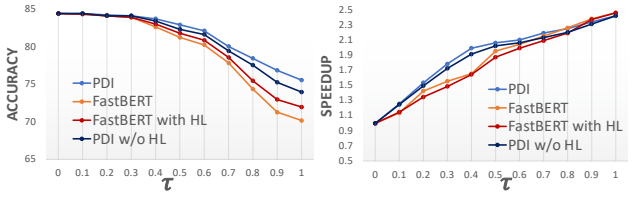

Fig. 3: Effect of contributing factors on the model performance for the QQP dataset.

Fig. 3 shows the results of this ablation study in terms of classification accuracy and model speedup. Based on the results, pyramid structure contributes more to the performance of the model. Using only the self-distillation loss can be beneficial in the cases where class labels are not available. However, the hybrid loss brings more accuracy and speedup to the model. Moreover, the hybrid loss can always be changed to the self-distillation loss by setting $\alpha = 0$.

4. CONCLUSION

In this paper we propose Pyramid Dynamic Inference (PDI) as a method for early routing in transformer-based architectures. PDI encourages fast inference by inserting strong classification heads at earlier layers of the network to boost the accuracy and certainty of the early routes. In order to prevent significant footprint increase and to leverage the semantic power of deeper transformer layers, PDI gradually reduces the complexity of the exit classifiers as we progress deeper into the network. Our experiments demonstrate promising results over GLUE benchmark where PDI outperforms the state of the art FastBERT on both accuracy and speedup over almost all tasks. Also, compared to the original 6-layer DistilBert, PDI achieves on average up to 3.38× speedup with 29% fewer parameters with only 2.5% accuracy degradation.We plan to extend this work to jointly train the exit classifiers and module replacing with a per-layer τ in a distillation setup. The proposed method can be used as a parallel strategy with other compression techniques.

5. REFERENCES

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv: 1810.04805*, 2018. 1

[2] A Radford, J Wu, R Child, D. Luan, Dario Amodei, and Ilya Sutskever, "Language models are unsupervised multitask learners," 2019. 1

[3] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R Salakhutdinov, and Q. V Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *NeurIPS*, 2019, pp. 5753–5763. 1

[4] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "Mobilebert: a compact task-agnostic bert for resource-limited devices," *arXiv:2004.02984*, 2020. 1

[5] A. Coucke et al., "Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces," *arXiv:1805.10190*, 2018. 1

[6] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," 2019. 1, 2

[7] Z. Wang, J. Wohlwend, and T. Lei, "Structured pruning of large language models," in *EMNLP*, 2020. 1

[8] C. Xu, W. Zhou, T. Ge, F. Wei, and M. Zhou, "BERT-of-theseus: Compressing BERT by progressive module replacing," in *EMNLP*, 2020, pp. 7859–7869. 1, 2

[9] Y. Gong, L. Liu, M. Yang, and L. D. Bourdev, "Compressing deep convolutional networks using vector quantization," *CoRR*, vol. abs/1412.6115, 2014. 1

[10] K. Zhen, H. D. Nguyen, R. Chinta, N. Susanj, A. Mouchtaris, T. Afzal, and A. Rastrow, "Sub-8-bit quantization aware training for 8-bit neural network accelerator with on device speech recognition," in *Interspeech 2022*, 2022. 1

[11] H. Duy Nguyen, A. Alexandridis, and A. Mouchtaris, "Quantization aware training with absolute-cosine regularization for automatic speech recognition," in *Interspeech 2020*, 2020. 1

[12] Xiaoqi J., Yichun Y., Lifeng S., Xin J., Xiao C., Linlin L., Fang W., and Qun L., "TinyBERT: Distilling BERT for natural language understanding," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Online, 2020, pp. 4163–4174. 1

[13] H. Saghir, S. Choudhary, S. Eghbali, and C. Chung, "Factorization-Aware Training of Transformers for Natural Language Understanding on the Edge," in *Proc. Interspeech 2021*, 2021, pp. 4733–4737. 1

[14] A. Acharya, R. Goel, A. Metallinou, and I. Dhillon, "Online embedding compression for text classification using low rank matrix factorization," in *AAAI*, 2019. 1

[15] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *Proc. Interspeech*, 2013, pp. 2365–2369. 1

[16] U. Thakker, J. G. Beu, D. Gope, G. Dasika, and M. Mattina, "Rank and run-time aware compression of NLP applications," *CoRR*, vol. abs/2010.03193, 2020. 1

[17] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," 2017. 1

[18] A. Tyagi, V. Sharma, R. Gupta, L. Samson, N. Zhuang, Z. Wang, and B. Campbell, "Fast intent classification for spoken language understanding," 2019. 1

[19] J. Xin, R. Tang, Y. Yu, and J. Lin, "BERxiT: Early exiting for BERT with better fine-tuning and extension to regression," in *EACL*, Online, 2021, pp. 91–104. 1

[20] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju, "Fastbert: a self-distilling bert with adaptive inference time," *arXiv:2004.02178*, 2020. 1, 2

[21] P. L Bartlett, N. Harvey, C. Liaw, and A. Mehrabian, "Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks," *JMLR*, vol. 20, no. 1, pp. 2285–2301, 2019. 1

[22] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018. 3

[23] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016. 3

[24] A. Williams, N. Nangia, and S. R Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," *arXiv preprint:1704.05426*. 3

[25] Alexis Conneau and Douwe Kiela, "Senteval: An evaluation toolkit for universal sentence representations," *arXiv preprint arXiv:1803.05449*, 2018. 3

[26] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D Manning, A. Y Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *EMNLP*, 2013, pp. 1631–1642. 3

[27] B. Dolan and C. Brockett, "Automatically constructing a corpus of sentential paraphrases," in *IWP*, 2005. 3