AT(G)	AT(B)	LevT(O)	LevT(T)
26.89	27.60	25.18	27.03

It	1	2	3	4	5	6	7	8	9	10	2.43
%	12.3	48.1	28.5	8.5	2.0	0.4	0.1	0	0	0.1	AVG

Table 1: (**Left**) Comparison of BLEU scores on the standard test set with Fairseq-py re-implementation; (**Right**) The percentage of test sentence generation terminated at each iteration using LevT(T) with a maximum iteration of 10.

We thank all the reviewers' insightful suggestions. We have revised the paper accordingly on language clarity and missing citations. For notations, we denote the standard Transformer as AT, and Levenshtein Transformer as LevT.

Confusion in Empirical Results (R1, R3): We have migrated our codebase to Fairseq, a popular sequence-to-sequence learning framework. We hope this provides more trusty numbers for comparisons between LevT models and baseline AT models. The updated results on WMT14 En-De are presented in Table 1, where G, B, O and T are short for greedy decoding, beam search (with beam size 4), oracle and teacher model, respectively. Fairseq implementation basically achieves improved results on both LevT and the baseline models, but has larger gaps between the two. As it stands, this still holds our conclusion on both performance and speed-up. We will update all other tables with the new implementation in the final version.

For summarization (Gigaword) experiments, we attribute the small gap between our AT baseline and the one R1 referred to (35.19/17.58/32.98 v.s. 35.51/17.35/33.17 for ROUGE-1/2/L) to implementation difference (e.g. OpenNMT has copy attention while ours does not). We will add SoTA numbers in the final version as R3 suggested. And we point out that our models lack task-specific architectures (e.g. pointer-generator decoder [5], which is effective and has become popular) and direct comparison with SoTA models is not fair.

Confusions on Iteration/Speed Comparison (R1, R3): We agree that injected noise in Figure 4 (a) is confusing, although it was originally for better visualization. We will remove it in the final version. We point out that our proxy for speed evaluation is the actual **machine execution time** [2, 4, 3], but not the number of iterations. We include the latter to show LevT's adaptive numbers of decoding iterations. And we clarify that a LevT iteration is not necessarily 3 times more expensive than an AT iteration (due to the "early exit" mechanism). We've updated Figure 4 (a) using the new implementation, and results on WMT14 En-De test set are presented in Table 1 (right). Most predictions are gotten in 1-4 iterations, and the mean is 2.43. Only a small portion ($\sim 0.1\%$) require the maximum number of iterations. From Figure 6 (in the Appendix), we see that the average number of iterations grows slowly with the sentence length.

Confusions on Model Architecture (R1, R2) A LevT decoding iteration contains one encoder forward pass and three decoder forward passes (for deletion, placeholder insertion and word filling). For the latter, we may not need to go through all the decoder layers, as described in Section 3.1 as "Early exit". In contrast, previous models require full passes of all the decoder layers [4].

Why learning from teacher is better than oracle? (R1, R3): First, we thank the R3 for pointing out the terminology issue. We will certainly stick with the community on the normal terms. As observed by many prior work [2, 4, 3], distillation from a teacher model reduces the complex modalities our dataset possesses, which is shown critical for any non-autoregressive (NAT) based models including the proposed LevT.

Imitation learning for baselines (R1): Teacher forcing is used as a standard technique to train AT models. From a high level standpoint, learning LevT falls into the same scope where the roll-out policy is from an expert, but the roll-in policy is a mixture. This is due to (i) lack of ground-truth path for target generation; (ii) the complementary insertion/deletion learning from the counterpart policy. Standard AT model already has the gold path provided by teacher forcing, so it is not necessary (and not possible) to apply the same algorithm to learn AT model.

Noisy Parallel Decoding (R3) As mentioned in Sec 3.3, our initial trial of using beam-search inside each prediction does not brings up much gain ($\approx +0.2$ BLEU). We conjecture similar results will happen in noisy decoding [1] due to its similar nature to beam search. We hypothesize this is because (i) LevT is able to revoke wrong predictions by deletion; (ii) the log-prob of LevT is not a good measure to select the best output. However, we do believe to see more improvements if we include an external re-ranker (e.g. AT teacher [2], language model). We will include this discussion and experiments in the final version.

References

[1] K. Cho. Noisy parallel approximate decoding for conditional recurrent language model. *arXiv preprint arXiv:1605.03835*.

[2] J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher. Non-autoregressive neural machine translation. *ICLR2018*.

[3] Ł. Kaiser, A. Roy, A. Vaswani, N. Parmar, S. Bengio, J. Uszkoreit, and N. Shazeer. Fast decoding in sequence models using discrete latent variables. *ICML2018*.

[4] J. Lee, E. Mansimov, and K. Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *EMNLP2018*.

[5] A. See, P. J. Liu, and C. D. Manning. Get to the point: Summarization with pointer-generator networks. *ACL2017*.