Method	Params	Valid	Test
NAS-RNN [42]	54M	-	62.40
AWD-LSTM [18]	24M	58.50	56.50
AWD-LSTM + FRAGE [8]	24M	58.10	56.10
AWD-LSTM + MoS [34]	22M	56.54	54.44
w/o dynamic evaluation			
ADV-AWD-LSTM [30]	24M	57.15	55.01
ADV-AWD-LSTM + *MaxUp*	24M	**56.25**	**54.27**
+ dynamic evaluation [13]			
ADV-AWD-LSTM [30]	24M	51.60	51.10
ADV-AWD-LSTM + *MaxUp*	24M	**50.83**	**50.29**

Table 9. Perplexities on the validation and test sets on the Penn Treebank dataset. Smaller perplexities refer to better language modeling performance. `Params` denotes the number of model parameters.

Method	Params	Valid	Test
AWD-LSTM [18]	33M	68.60	65.80
AWD-LSTM + FRAGE [8]	33M	66.50	63.40
AWD-LSTM + MoS [34]	35M	63.88	61.45
w/o dynamic evaluation			
ADV-AWD-LSTM [30]	33M	63.68	61.34
ADV-AWD-LSTM + *MaxUp*	33M	**62.48**	**60.19**
+ dynamic evaluation [13]			
ADV-AWD-LSTM [30]	33M	42.36	40.53
ADV-AWD-LSTM + *MaxUp*	33M	**41.29**	**39.61**

Table 10. Perplexities on the validation and test sets on the WikiText-2 dataset. Smaller perplexities refer to better language modeling performance. `Params` denotes the number of model parameters.

.1. Language Modeling

For language modeling, we test *MaxUp* on two benchmark datasets: Penn Treebank (PTB) and Wikitext-2 (WT2). We use the code provided by [30] as our baseline[4], which stacks a three-layer LSTM and implements a bag of regularization and optimization tricks for neural language modeling proposed by [18], such as weight tying, word embedding drop and Averaged SGD.

For this task, we apply *MaxUp* using word embedding dropout [18] as the random data augmentation method. Word embedding dropout implements dropout on the embedding matrix at the word level, where the dropout is broadcasted across all the embeddings of all the word vectors. For the selected words, their embedding vectors are set to be zero vectors. The other word embeddings in the vocabulary are scaled by $\frac{1}{1-p}$, where p is the probability of embedding dropout.

As the word embedding layer serves as the first layer in a neural language model, we apply *MaxUp* in this layer. We do feed-forward for m times and select the worst case to do backpropagation for each given sentence. In this section, we set a small $m = 2$ since the models are already well-regularized by other regularization techniques.

Implement Details The PTB corpus [17] is a standard dataset for benchmarking language models. It consists of 923k training, 73k validation and 82k test words. We use the processed version provided by [19] that is widely used for PTB.

The WT2 dataset is introduced in [18] as an alternative to PTB. It contains pre-processed Wikipedia articles, and the training set contains 2 million words.

The training procedure can be decoupled into two stages: 1) optimizing the model with SGD and averaged SGD (ASGD); 2) restarting ASGD for fine-tuning twice. We apply *MaxUp* in both stages, and report the perplexity scores at the end of the second stage. We also report the perplexity scores with a recently-proposed post-process method, dynamical evaluation [13] after the training process.

[4]`https://github.com/ChengyueGongR/advsoft`

Results on PTB and WT2 The results on PTB and WT2 corpus are illustrated in Table 9 and Table 10, respectively. We calculate the perplexity on the validation and test set for each method to evaluate its performance. We can see that *MaxUp* outperforms the state-of-the-art results achieved by Frage [8] and Mixture of SoftMax [34]. We further compare *MaxUp* to the result of [30] based on AWD-LSTM [18] at two checkpoints, with or without dynamic evaluation [13]. On PTB, we enhance the baseline from $55.01/51.10$ to $54.27/50.29$ at these two checkpoints on the test set. On WT2, we enhance the baseline from $61.34/40.53$ to $60.19/39.61$ at these two checkpoints on the test set. Results on validation set are reported in both Table 9 and 10 to show that the improvement can not achieved by simple hyper-parameter tuning on the test set.