Training RNNs As Fast As CNNs

Breandan Considine
260815673
breandan.considine@mail.mcgill.ca

Rajveer Gandhi
260719747
rajveer.gandhi@mail.mcgill.ca

Muhammed Safa
260348556
muhammed.safa@mail.mcgill.ca

Link To Repository: https://github.com/msalihs/sru

***Abstract*— In this paper we will be reproducing the results of the Simple Recurrent Architecture (SRU) which was proposed by Tao Lei et al [1] as part of their submission to ICLR in the year 2018. The proposed SRU architecture is a recurrent unit that simplifies the computation in order to increase parallelism, enabling us to obtain much faster training/test time. This is in contrast to the recurrent neural network which is a powerful tool for sequence modeling tasks, but the dependence of its computation at each timestep depending on the output of the previous timestep puts limitations on parallelization, leading to expensive results for very long sequences. We will be reproducing the results of the authors for a variety of tasks i.e classification, language modeling and speech recognition in order to demonstrate the effectiveness of the SRU and reaffirm it as a viable choice for tasks relating to Natural Language Processing.**

I. INTRODUCTION

Recurrent Neural Networks (RNNs) is a common architecture used to work with sequential data which depends on previous inputs where information is passed on from timestep to timestep through the hidden layers. This quality makes RNNs a conspicuous target for NLP tasks. However, the sequential dependency of the RNNs limit any potential to parallelize our computations. Possessing the ability to process sequences of data and retaining the temporal order of the symbols, these advantages follow with one caveat; the computation at each step depends on the previous step. This makes RNNs less accommodating to parallelization.

Vanilla RNNs also have two troubling issues, vanishing/exploding gradients and the inability to preserve long term dependencies. The LSTM is a special kind of RNN which solves these two problems [2]. LSTM uses structures called gates in order to remove or add any information. These gates may consist of an activation function (i.e sigmoid) and a point-wise multiplication operation. But due to the intricacies of these gates, training time is vastly increased.

At the heart of pattern recognition problems like computer vision and voice recognition lies the convolutional neural network (CNN). We feed the output of each convolutional layer to the next convolutional layer. Even though this architecture is more often used for tasks that involve image data, they have also been used for sequence modeling[3]. Some notable vantages of CNNs is that they allow for increased scalability to very long sequences and increased parallelism. But as sequence processing assume time invariance, convolutional neural networks do not work well with large scale sequence modeling.

For this paper, we will be reproducing the results obtained by Tao Lei et al with their proposed Simple Recurrent Architecture (SRU). In this architecture, we offer to do the majority of the computation at each time step without depending on the previous time steps computation. This allows us to parallelize the heavy computations. We will then use element-wise operations to do lightweight computations on the recurrent combination [Fig. 1. [1]]. This architecture allows us to address drawbacks of standard models like CNNs and RNNs. Like CNNs, SRU allow for parallel computations across time steps, making them scalable to long sequences. Like RNNs, SRU allows the output to depend on the sequence of elements and the temporal order of these elements. We will test the SRU architecture on classification and a variety of natural language processing tasks like question-answering, language modeling in order to reproduce the results of the authors wherein the SRU strongly outperformed the LSTM baselines for the same tasks whilst reducing computation time.

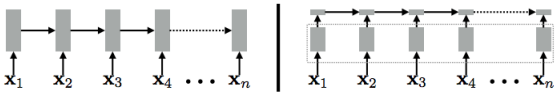

Fig. 1. Illustration of the difference between RNNs (left) and the SRU architecture (right). On the left, the entire computation (the gray block) depends on previous time steps computations, making it hard to parallelize our computations. For our SRU architecture, we will be doing our heavy computations independent of the other time steps (larger gray blocks) allowing us to parallelize a majority of the computations (surrounded by the dashed lines), and taking care of the recurrent combinations with relatively lightweight computation (small gray blocks).

II. THE MODEL

In structures like LSTM's, we discussed how we use structures called gates in order to regulate information flow. The authors define a gate to be composed of a single-layer feed-forward network with a sigmoid activation. A point-wise multiplication operation is then done on the inputs using the gate output. The computation of the feed-forward network consists of a matrix multiplication which is very expensive compared to the point-wise multiplication that combines two inputs (the current and previous time stamp) which is relatively lightweight. The model design for SRU takes into account one key decision; making the gate computation dependent only on the current input. The matrix multiplications involved in this computation can then be easily parallelized, leaving us with only the point-wise

multiplication computation which depends on the previous time steps.

The basic form of SRU includes a single forget gate. Given an input x_t at time t, we compute a linear transformation $\tilde{x}_t$ [4][5] and the forget gate f_t:

$$\tilde{x}_t = \mathrm{W}x_t$$
$$f_t = \sigma(W_f x_t + b_f)$$

These computations depend only on x_t and can be easily parallelized. The forget gate is used to model the internal state c_t, which is used to compute the output state h_t:

$$c_t = f_t \odot c_{t-1} + (1\text{-}f_t) \odot \tilde{x}_t$$
$$h_t = \mathrm{g}(c_t)$$

where $\mathrm{g}(\cdot)$ is an activation function (a sigmoid function in this case, as used by the authors) to compute the output state h_t. The complete architecture as used by the authors includes skip connections[6][7], highway connections[6] and a reset gate which is used to compute the output state h_t as a combination of the internal state $\mathrm{g}(c_t)$ and the input x_t:

$$\tilde{x}_t = \mathrm{W}x_t \quad (1)$$
$$f_t = \sigma(W_f x_t + b_f) \quad (2)$$
$$r_t = \sigma(W_r x_t + b_r) \quad (3)$$
$$c_t = f_t \odot c_{t-1} + (1\text{-}f_t) \odot \tilde{x}_t \quad (4)$$
$$h_t = r_t \odot \mathrm{g}(c_t) + (1\text{-}r_t) \odot x_t \quad (5)$$

In this architecture the computation bottleneck is the matrix computations in equations 1-3, which can be computed in parallel. The recurrent computations are equations 4-5 which involve element-wise multiplication and are much faster to compute. The authors apply two optimizations to the SRU architecture which we have used as well:

- Matrix multiplications across all time steps are batched to reduce computation intensity and GPU utilization.
- All element-wise operations of the sequence are fused into one kernel function so that operations such as addition and sigmoid activation don't invoke a separate function call at each step.

III. RELATED WORK

The approach of the authors is to integrate the recent work done by Lei et al[9][11] on recurrent convolutions, Kernel networks[4] and Quasi-RNN. Quasi-RNN is an approach to neural sequence modeling that alternates convolutional layers, which apply in parallel across each timestep, and a minimalist recurrent pooling function that applies in parallel across channels[10]. The SRU architecture used by the authors is largely related to a QRNN with a convolutional window size of one, with both architectures aiming to speed up the computation time of recurrent neural networks. Even with the strong similarities between the two, the SRU architecture has largely been evolved as an instance of the recurrent architectures introduced by Lei et al[11]. The results from [4] has helped the authors in deriving the kernels for the neural operations in their SRU architecture.

IV. EXPERIMENTS

We reproduce the results of the authors with tasks like classification, question answering, language modeling and speech recognition in order to reevaluate the SRU architecture. These experiments were performed on an Intel x86_64 (Haswell architecture) and uses a NVIDIA Tesla K80 GPU. The model configurations have been set identical to prior work by the authors.

Author's source code was publicly available and used as is for the most part[1]. Any modification or changes that needed to be performed is described in sections below.ss

A. CLASSIFICATION

Dataset SRU is evaluated with text classification task on 6 different datasets : movie review sentiment (MR)[13], subjectivity (SUBJ)[14], customer reviews polarity (CR)[15], TREC question type (TREC)[16], MPQA opinion polarity (MPQA)[17], and the Stanford sentiment treebank (SST)[18]. The same datasets are used in Kim et al. to evaluate the performance of their CNN architecture [12].

In the paper, authors use *word2vec* embeddings which is pre-trained on 100 billion Google News tokens. Due to hardware and time constraints, we use *LexVec* which is pre-trained on English Wikipedia articles [2]. Although much smaller in size, it performs relatively well and yields very similar results as will be shown in the following sections.

Setup We keep the same methodology as the paper and train RNN encoders using the last output state to predict the class label for a given sentence. Two-layer RNN encoder with 128 hidden dimensions is used for LSTM models with the exception of SST dataset. For this dataset, a four-layer RNN is used to accommodate for the richer content. CNN model is based on the one presented in Kim et al. [12] with the same filter windows of 3, 4, and 5. Adam optimizer is used with default 0.001 learning rate and 0 weight decay. Authors do not specify the dropout probability that they use in their paper, therefore it is selected based on the result of a cross validation among values of $\{0.1; 0.3; 0.5; 0.7\}$. The result of this hyper-parameter tuning is presented in Appendix A.

Finally, each model is trained for 100 epochs while performing 10-fold cross validation when no standard split is specified. Each model is trained 5-times on each of the datasets and the average is computed to find the overall test accuracy.

Results The results are compared in Table I and are similar to the ones presented in the paper overall. We observe that the LSTM is slightly off for MR and CR datasets in particular but otherwise our results are within 1.5% of what is presented by

[1] https://github.com/taolei87/sru

[2] https://github.com/alexandres/lexvec

	Model	CR	SUBJ	MR	TREC	MPQA	SST
Presented results	CNN [12]	82.2 ± 2.2	92.9 ± 0.7	79.1 ± 1.5	93.2 ± 0.5	88.8 ± 1.2	85.3 ± 0.4
	LSTM	82.7 ± 2.9	92.4 ± 0.6	80.3 ± 1.5	93.1 ± 0.9	89.2 ± 1.0	87.9 ± 0.6
	SRU	84.8 ± 1.3	93.4 ± 0.8	82.2 ± 0.9	93.9 ± 0.6	89.7 ± 1.1	89.1 ± 0.3
Reproduced results	CNN [12]	83.2 ± 0.8	93.7 ± 0.3	80.7 ± 0.6	92.9 ± 0.5	88.8 ± 0.2	84.5 ± 0.4
	LSTM	79.5 ± 1.0	93.1 ± 0.5	77.5 ± 1.0	93.8 ± 0.5	88.9 ± 0.5	86.2 ± 0.6
	SRU	82.4 ± 1.1	94.1 ± 0.5	81.2 ± 0.6	94.0 ± 0.4	90.3 ± 0.3	87.1 ± 0.2

TABLE I

TEST ACCURACY ON ALL 6 DATASETS AND COMPARISON OF REPRODUCED RESULTS AND RESULTS PRESENTED IN THE PAPER

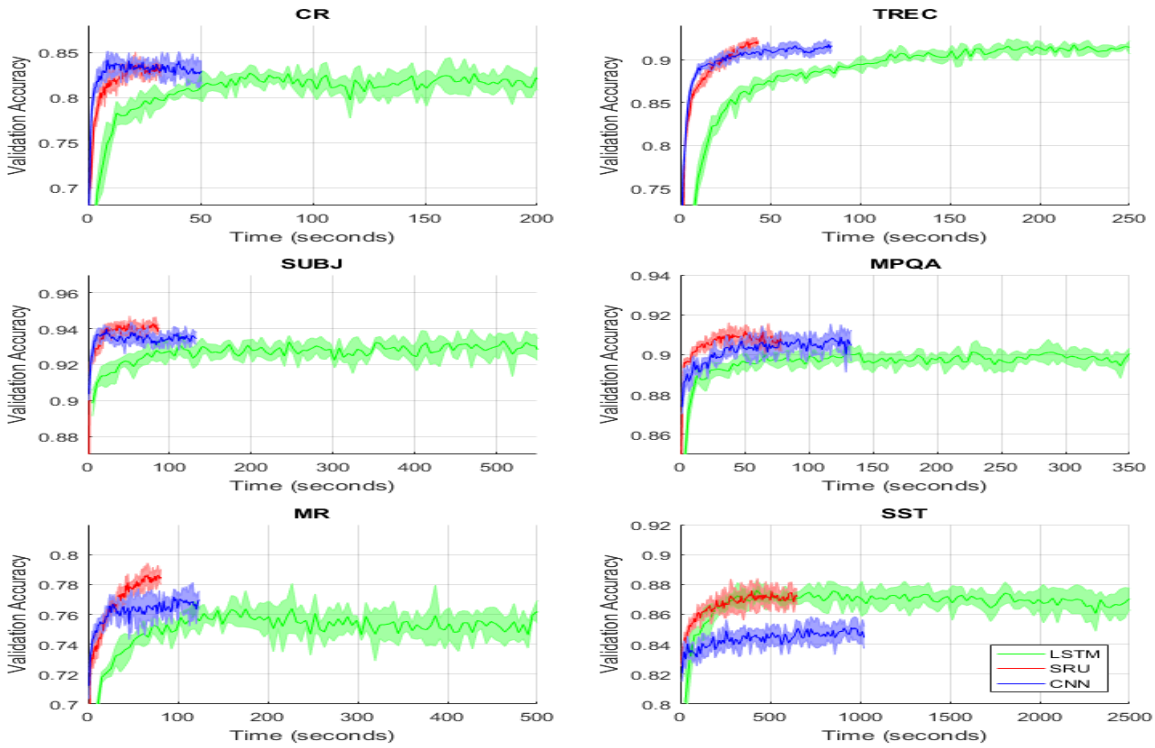

Fig. 2. Mean validation accuracies vs training time (in seconds) of LSTM, CNN, and SRU for the first 100 epochs on the six classification tasks.

the authors. From this table, it is also clear that SRU yields equal or better results than LSTM in all classification tasks.

Figure 2 shows the validation accuracy for 100 epochs of training for each dataset and architecture as well as how long it takes to complete 100 epochs. SRU not only yields better test accuracies but training time is around 6x faster in every classification task when compared to LSTM based RNN architecture.

B. QUESTION ANSWERING

Dataset SRU is evaluted on Standford Question Answering Dataset (SQuAD) [19]. It is a reading comprehension dataset, consisting 100,000+ question-answer pairs on 500+ Wikipedia articles. Dataset is made up of questions posed by crowdworkers on a set of Wikipedia articles and the answer to each question is a segment of text from the corresponding reading passage. Both the standard train and development sets are used during evaluation.

Setup LSTM and SRU based RNN models are compared in performance using Document Reader model [20]. An open-source implementation of this model is used for evaluation. Authors note that it performs 1% compared to the reported results. In order to improve the performance, they used a learning rate of 0.001 instead of 0.002, Adamax optimizer, and seperately tuned dropout rates for RNN layers and word embeddings. We use a dropout rate of 0.5 for input word-embeddings, 0.2 for SRU layers and 0.3 for LSTM layers. All models are trained up to 30 epochs, batch-size 32 and a hidden dimensionality of 128.

Results Results are presented in Table II. For the bi-directional LSTM based RNN, we achieve an exact match of 66.2% and 75.4 F1 score. Similarly, we get 67.6% exact

	Model	# layers	d	Dev EM	Dev F1	Time per epoch
Presented results	Chen et al. [20]	3	128	69.5	78.8	-
	Bi-LSTM	4	128	69.6	78.9	872s
	Bi-SRU	4	128	69.7	79.1	193s
Reproduced results	Bi-LSTM	4	128	66.2	75.4	1206s
	Bi-SRU	4	128	67.9	77.8	322s

TABLE II

EXACT MATCH (EM) AND F1 SCORES OF VARIOUS MODELS ON SQUAD DATASET. THE TABLE ALSO INCLUDES A COMPARISON OF TOTAL PROCESSING TIME PER EPOCH FOR EACH MODEL.

	Model	Layers	Size	Dev PPL	Test PPL	Time	Epoch
Presented results	cuDNN LSTM	2	24m	73.3	71.4	53s	73s
	cuDNN LSTM	3	24m	78.8	76.2	64s	79s
	SRU	3	24m	68.0	64.7	21s	44s
	SRU	4	24m	65.8	62.5	23s	44s
	SRU	5	24m	63.9	61.0	27s	46s
	SRU	6	24m	63.4	60.3	28s	47s
Reproduced results	cuDNN LSTM	2	24m	76.80	73.15	55s	76s
	cuDNN LSTM	3	24m	78.8	81.58	67s	82s
	SRU	3	24m	69.31	65.13	22s	44s
	SRU	4	24m	66.46	62.73	23s	43s
	SRU	5	24m	65.16	61.33	26s	45s
	SRU	6	24m	64.17	60.66	29s	48s

TABLE III

LANGUAGE MODEL RESULTS. PRESENTED RESULTS WERE REPRODUCIBLE WITHIN A SMALL MARGIN OF ERROR.

match and 77.3 F1 score using the bi-directional SRU based RNN. The results for SRU are very similar to that of authors with around 1.5% difference. This difference can be explained by the fact that we are only able to run the models for 30 epochs instead of 50 epochs due to time and hardware constraints. For the benchmark LSTM model, we obtain a 3% lower than both the baseline authors were able to achieve as well as what was reported in the original paper from Chen et al[20]. It is most likely due to a non-optimized dropout value or learning rate in our training. But overall, we are able to confirm that the SRU model outperforms the LSTM model in both the exact match and F1 score metrics.

More importantly, Tao et al.'s SRU achieves a significantly faster training time while achieving better accuracy in exact match and F1 scores. The authors presented a 69% reduction in training time with SRU when compared to LSTM. This result aligns with the 71% reduction in training time that we observe in our results.

C. LANGUAGE MODELING

Dataset The Penn Treebank (PTB) is a standard dataset for language modeling, consisting of 2,499 articles from the Wall Street Journal, annotated with syntatic parts of speech. The raw text data is commonly used to train language models, and can easily be obtained from the official website [3], or on GitHub [4] as suggested by the authors. The original train, test and development (ie. validaton) sets from Mikolov et al. [21] are used during evaluation.

Setup Experimental setup as described was reproducible using the published source code. The authors compare four SRU networks across hyperparameter settings, varying layer depth and width, and two variations of network depth using the cuDNN LSTM implementation from CuPY version 4.0.

Results All results were reproducible with a small error (approximately 1% higher on all reported metrics). Our performance on the default hyperparameter settings (6 layers deep, 910 units wide), yielded a test and validation perplexity of 60.66 and 64.17 respectively. See Table III for full experimental results of our language model experiment.

D. SPEECH RECOGNITION

Dataset The Switchboard-1 corpus for labeled speech [22] is composed of 260 hours of recorded English speech, containing 2,400 telephone conversations between unacquainted adults. While considerably smaller than recent datasets, it is a sufficiently large benchmark for comparison, with lower overhead for reproducing experimental results.

Setup The authors use a forked version of CNTK with a latency-controlled bidirectional LSTM (LC-BLSTM) custom implementation, and Kaldi[23], a standard library for speech research. While the full source code was published, the instructions provided on the authors' GitHub repository [5] were insufficient to build the project dependencies, and we were unable to reproduce the speech experiment as described, despite the author's helpful assistance[6].

[3]https://catalog.ldc.upenn.edu/ldc99t42

[4]https://github.com/yoonkim/lstm-char-cnn/tree/master/data/ptb

[5]https://github.com/taolei87/sru/blob/master/speech/README.md

[6]https://github.com/taolei87/sru/issues/36

Results The authors' published results are generated using a forked version of the CNTK deep learning framework, which we were not able to recompile due to undocumented software issues. The final experimental results were not reproducible due to the aforementioned technical obstacles encountered during setup.

V. CONCLUSIONS

We conclude that Tao et al.'s SRU architecture offers important advantages for parallelism and scaling, facilitating the training of recurrent neural networks on larger datasets with commodity hardware. It achieves higher accuracy in the same number of epochs as traditional LSTMs, using less wall clock time. Our results infer that this paper does what it proposes; train RNNs as fast as CNNs while being accurate and efficient on large tasks. We believe that SRU can be an efficient architecture for NLP tasks, and an effective substitute for LSTM.

REFERENCES

[1] Tao Lei, Yu Zhang, Yoav Artzi. *Training RNNs as Fast as CNNs.* 2018

[2] Ivan Aguilar, Brandon McKinzie. *RNNs, LSTMs, and Reddit Comments.* 2015

[3] Xiang Zhang, Junbo Zhao, and Yann LeCun. *Character-level convolutional networks for text classification.* 2015.

[4] Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. *Deriving neural architectures from sequence and graph kernels.* ICML, 2017.

[5] Kenton Lee, Omer Levy, and Luke Zettlemoyer. *Recurrent additive networks.* arXiv preprint arXiv:1705.07393, 2017.

[6] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutnk, Bas R Steunebrink, and Jrgen Schmidhuber. *Lstm: A search space odyssey.* arXiv preprint arXiv:1503.04069, 2015

[7] Huijia Wu, Jiajun Zhang, and Chengqing Zong. *An empirical exploration of skip connections for sequential tagging.* In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, December 2016.

[8] Tao Lei, Regina Barzilay, and Tommi Jaakkola. *Molding cnns for text: non-linear, non-consecutive convolutions.* In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2015.

[9] Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Kateryna Tymoshenko, Alessandro Moschitti, and Llus Mrquez. *Semi-supervised question retrieval with gated convolutions.* In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2016.

[10] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher.

[11] Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Deriving neural architectures from sequence and graph kernels. ICML, 2017. *Quasi-recurrent neural networks.* In ICLR, 2017

[12] Yoon Kim. *Convolutional neural networks for sentence classification.* In Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014), 2014.

[13] Bo Pang, Lillian Lee. *Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales.* In Proceedings of the 43rd annual meeting on association for computational linguistics, pp. 115124. Association for Computational Linguistics, 2005.

[14] Bo Pang, Lillian Lee. A sentimental *education: Sentiment analysis using subjectivity summarization based on minimum cuts.* In Proceedings of the 42nd annual meeting on Association for Computational Linguistics, pp. 271. Association for Computational Linguistics, 2004.

[15] Minqing Hu, Bing Liu. *Mining and summarizing customer reviews.* In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 168177.ACM, 2004.

[16] Xin Li,Dan Roth. *Learning question classifiers.* In Proceedings of the 19th international conference on Computational linguistics-Volume 1. Association for Computational Linguistics, 2002.

[17] Janyce Wiebe, Theresa Wilson, and Claire Cardie. *Annotating expressions of opinions and emotions in language.* Language resources and evaluation, 2005.

[18] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. *Recursive deep models for semantic compositionality over a sentiment treebank.* In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 16311642, October 2013.

[19] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. *Squad: 100,000+ questions for machine comprehension of text.* In Empirical Methods in Natural Language Processing (EMNLP), 2016.

[20] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. *Reading Wikipedia to answer open-domain questions.* In Association for Computational Linguistics (ACL), 2017.

[21] Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. *Recurrent neural network based language model.* In INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010, pp. 10451048, 2010.

[22] J. J. Godfrey, E. C. Holliman, and J. McDaniel. Switchboard: Telephone speech corpus for research and development. In Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 517520, 1992.

[23] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na,Yiming Wang, and Sanjeev Khudanpur. Purely sequence-trained neural networks for asr based on lattice-free mmi. In INTERSPEECH, pp. 27512755, 2016.

APPENDIX

A. *Cross-validation results for drop out rate*

We perform a cross-validation to tune dropout rate. The results are presented below in Table IV.

Model	Dropout	CR	SUBJ	MR	TREC	MPQA	SST
SRU	0.1	0.222	0.088	0.267	0.073	0.111	0.14
SRU	0.3	0.214	**0.081**	**0.243**	0.068	**0.111**	**0.139**
SRU	0.5	**0.209**	0.089	0.275	0.08	0.117	0.15
SRU	0.7	0.231	0.093	0.294	**0.065**	0.114	0.18
CNN	0.1	**0.185**	**0.063**	0.211	0.075	**0.116**	0.166
CNN	0.3	0.192	0.069	0.203	0.081	0.12	**0.154**
CNN	0.5	0.207	0.08	**0.178**	**0.07**	0.121	0.163
CNN	0.7	0.224	0.086	0.235	0.085	0.128	0.191
LSTM	0.1	0.215	0.08	0.235	0.076	0.11	0.144
LSTM	0.3	0.218	0.081	**0.227**	0.076	**0.108**	0.136
LSTM	0.5	**0.21**	**0.07**	0.253	**0.072**	0.11	**0.132**
LSTM	0.7	0.222	0.078	0.291	0.078	0.13	0.142

TABLE IV

THIS TABLE SHOWS THE ERROR RATE FOR EACH OF THE TEST FOR A GIVEN DROPOUT RATE. A DIFFERENT DROPOUT IS SELECTED FOR EACH MODEL AND DATASET BASED ON THE LOWEST ERROR OBSERVED DURING CROSS VALIDATION. LOWEST ERROR RATES ARE HIGHLIGHTED FOR A GIVEN TEST AMONG DROPOUT VALUES OF $\{0.1, 0.3, 0.5, 0.7\}$