Faster Transformers for Document Summarization

Dian Ang Yap
Electrical Engineering
Stanford University
dayap@stanford.edu

Vineet Kosaraju
Computer Science
Stanford University
vineetk@stanford.edu

Zaid Nabulsi
Computer Science
Stanford University
znabulsi@stanford.edu

Abstract

The field of sequence transduction has been dominated by complex networks with an encoder/decoder structure. In recent years, the best performing models for various tasks, such as text summarization and machine translation, have included an attention mechanism. The current state of the art is obtained by transformers (19), but these models are inefficient and intractable with long inputs. In this paper, we aim to experiment with design architectural improvements to attentions in thee encoder that improves performance in long inputs but also able to achieve results comparable to the state of the art. We introduce two novel architectural changes: a multi-head compressed attention module that groups words using convolutions, and a strided neighborhood attention that relaxes and reduces long term dependencies. We then apply these models to large document summarization and show empirically a speedup of 5.5% in training and 6.8% in inference time, while obtaining better ROUGE-2 precision and F1 scores, suggesting that our architectures may be successfully applied to other language understanding tasks.

1 Introduction

In recent years, there have been several successful innovations with neural networks that have improved the field of natural language processing. Chief of these innovations are recurrent architectures, such as RNNs (6), which allow for multi-word sequence encodings, and attention mechanisms (19), which improve the accuracy of these recurrent systems. The main drawback of recurrent systems is that they require sequential processing of the input. Transformers aimed to mitigate this limitation of RNNs by replacing the recurrent networks with feedforward layers. Transformers have become highly successful on a wide variety of tasks such as machine translation (19), document summarization (9), language modeling (14), and question answering (4). However, the main drawback with transformers is that their attention layers serve as bottlenecks. Specifically, the attention of transformers is quadratic in performance with respect to the input length: $O(\text{len}(\text{seq})^2)$. Thus, this motivates us to design architectural improvements to transformers for more efficient training, while maintaining comparable accuracy to the existing state-of-the-art methods on document summarization.

2 Related Work

Tasks involving sequence transduction, including, including document summarization, have previously been attempted through various means. Graves (8) and Chopra et. al. (2) try using recurrent neural networks; Keneshloo et. al. (10) employs a wide variety of RL models for abstractive text summarization; Dutil et. al. (5) and Rush et. al. (15) introduce sequence-to-sequence models using attention, and Li et. al. (11) use a generator and evaluator in order to generate paraphrases.

More popular in literature has been the use of attention for summarization as a sequence to sequence tasks. In fact, Medina, Bahuleyan and Shi et. al. all employ a type of attention specifically for a machine transduction task, with the the latter focusing on text summarization (13), (1), (17). Xu et.

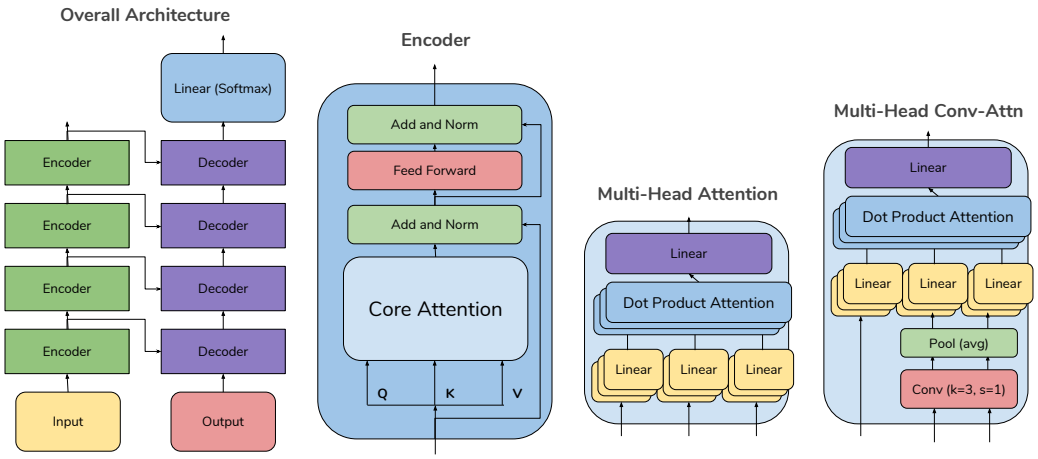

Figure 1: The transformer consists of several encoder (green) and decoder (purple) blocks (left). Inside each encoder block is a core attention module that is computed on a given query, key, and value sentence. The baseline transformer uses a multi-head attention module here, which computes self-attention multiple times in parallel. In Section 3.3, we propose an augmented convolutional attention module that groups words into phrases and reduces the operations performed.

al. introduces graphs on top of attention (20), and Vaswani introduces the transfomer (19) which was extended by Gehrmann who uses bottom-up attention to summarize source texts with a maximum capped length of 400 (7). Dai et. al. builds on transformers and introduces a new novel architecture intended to model long-range dependencies, known as the Transformer-XL (3).

However, while these papers present near state-of-the-art results for source texts at a capped length, the attention module lacks performance accuracy and speedups in even longer sequences, due to the bottleneck from the attention module which jeopardizes performance and efficiency. Moreover, modelling long sequences efficiently is a relatively new field, with Liu et. al. introducing memory-compressed and local attention in a decoder-only module for summarizing sources to generate Wikipedia articles (9). Here, we introduce two new attention modules that, when used in the encoder, allow the transformer to perform well for long sequences of source text.

3 Approach

3.1 Task Definition

The specific task we are focused on is that of automatically summarizing long sentences, paragraphs, and even entire documents. As mentioned, summarization is one example of several goals in NLP where being able to process large amounts of text efficiently is crucial, due to the long input sequences. We utilize the CNN/Dailymail Summarization dataset (Section 4.1) and focus on summarizing a full document into a single paragraph. Notation wise, for a single example we map the raw text, $(x_1, x_2, \ldots, x_{m'})$, to a summarized version, $(y_1, y_2, \ldots, y_{n'})$, where $m' >> n'$ and $m', n' \in \mathbb{Z}$, with x_i, y_j as word tokens.

3.2 Transformers

For our baseline architecture we chose a vanilla transformer, which comprises of several layers of encoder and decoder modules. As opposed to using a recurrent structure to sequentially process words, the transformer parallelizes the computation using several linear (fully connected) layers and self-attentive layers (Figure 1). Within each encoder/decoder is also residual skip-connections to aid the learning process.

Each attention layer is slightly more complex and takes in three values: a value (V), a key (K), and a query (Q). From these values, it applies a standard dot-product attention, as shown in Equation 1, with a scaling factor of $\frac{1}{\sqrt{d_k}}$ added to ensure that the value of the dot product doesn't grow too large

Table 1: Comparison of runtime, number of non-parallelizable operations, and length of dependencies supported for a recurrent architecture compared to the transformer architecture.

Layer	**Asymptotic Runtime**	**Sequential Operations**	**Dependency Length**
RNN (Recurrent)	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Transformer (Baseline)	$O(n^2 \cdot d)$	$O(1)$	$O(n)$

in magnitude with respect to d_k, the dimension of the key.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

However, instead of running self-attention on an inputted key, query, and value only once, the network instead makes use of a module called multi-head attention. This module repeatedly runs attention using different weights on the same inputted triplet. The benefit here is that multiple attentive properties can be learned in a parallel fashion, essentially allowing for ensembling of attention weights and learning of a full distribution of word importance. Generally, we define $h = 8$ unique heads, where each head has its own learnable weights, ${W_i}^Q$ for the query, ${W_i}^K$ for the key, and ${W_i}^V$ for the value, and the heads are combined using an overall set of learnable weights, W^O:

$$\begin{aligned} \text{head}_i &= \text{Attention}(Q{W_i}^Q, K{W_i}^K, V{W_i}^V), \\ \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \end{aligned} \tag{2}$$

3.3 Proposed Models

Even though the core transformer model is very promising and improves upon recurrent architectures, it does have some limitations. Consider Table 1, where we compare the transformer model's performance in terms of asymptotic runtime, number of operations, and dependency length. Note that n represents the length of the sequence, and d represents the embedding hidden dimension.

While it is true that the transformer improves over a recurrent architecture by reducing the number of sequential operations from n to 1, this improvement does have some limitations. Specifically, the asymptotic runtime per operation of the transformer is only faster than recurrence when $n < d$. This is generally true in NMT tasks, where translated sentences are relatively short, but is not true in summarization, where n is often the entire length of a document.

The main reason this runtime is large in the transformer is that it is supporting a maximum dependency length of $O(n)$: in other words, the very first word can have dependencies on the very last word. Again, this is useful in NMT, but is not as necessary in document summarization, where we often want to summarize paragraphs individually into shorter sentences.

As such, in our proposed changes we hope to decrease the runtime of each operation, while maintaining $O(1)$ operations, by leveraging the reduced need for long dependencies in document summarization tasks. Even though long term dependencies are still necessary for true natural language understanding, we believe long document summarization can be approximated with partial language understanding from shorter dependencies. Note that our proposed changes are only performed in the encoder self-attention, as that remains the bottleneck of the network.

3.3.1 Convolutional Model

As the dot product attention remains the largest bottleneck in the transformer architecture, we wanted to reduce the number of computations it required. We introduce a modification of the multi-headed attention module that shortens the key and value by convolving over sequence lengths beforehand:

$$\text{MultiHeadConv}(Q, K, V) = \text{MultiHead}(Q, \text{Conv1D}(K^\top)^\top, \text{Conv1D}(V^\top)^\top) \tag{3}$$

The convolution groups surrounding words into phrases, and using a nonzero stride reduces the size of the input. We propose two ways of incorporating this change into the multi-headed attention: directly using the convolution with a stride equal to the kernel size, or by convolving with a stride of one,

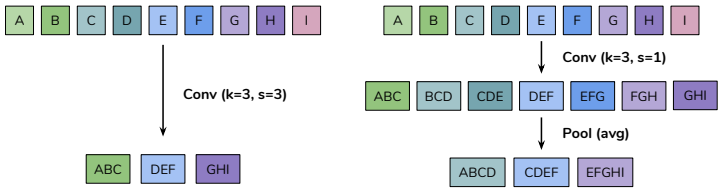

Figure 2: A comparison of the receptive fields of the convolution-only model, and the convolution with pooling model shows that only the latter explicitly groups all neighboring words into phrases.

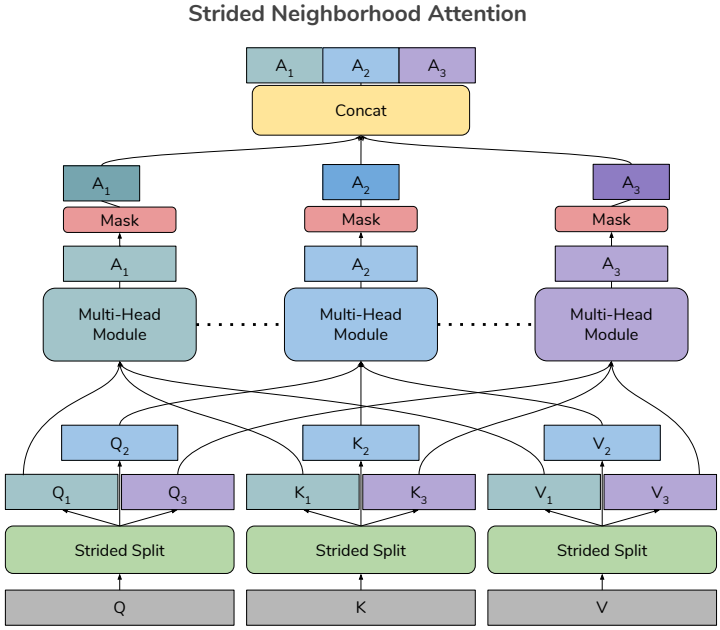

Figure 3: We also propose a strided neighborhood attention module that takes the place of the core attention module in the transformer encoder. Each query, key, and value is split into overlapping (strided) blocks. Each corresponding split of the queries, keys, and values is processed with a multi-head attention module (either the original, or our proposed convolutional one), which generates an attention output A_i. These outputs are masked and concatenated together to form the final attention module output. Note that the multi-head attention modules share weights.

and then pooling afterwards (Figure 1). While these two proposed models are related, they perform very different computations. Consider Figure 2, which compares the receptive fields of these two models on a contrived input sequence. While the convolution only model groups words into phrases, there is no overlap between phrases, and so surrounding words are not explicitly grouped together. Alternatively, with the model that also uses pooling, neighboring words are always explicitly grouped together. As such, we expect the latter model to outperform the prior.

3.3.2 Strided Neighborhood Attention

As discussed earlier, with document summarization we hope to relax the requirement of full-document dependencies to reduce computation requirements. Local attention, as proposed by Liu et. al. (9), addresses this dependency relaxation by splitting the input into blocks, and running attention independently on each block. However, the problem with this approach is that there are no interactions between the blocks, and so both short and long term dependencies are lost. To address this flaw, we implement a novel strided neighborhood attention model that like local attention, restricts attention dependencies to blocks. However, unlike local attention, blocks overlap so that neighboring words

may still interact within a layer. Further, while this reduces dependencies within each layer, over enough layers, distant words still interact, ensuring the presence of long-term dependencies. This strided neighborhood attention takes the place of the core attention module in the encoder in Figure 1. Our proposed model has four main steps: 1) strided split, 2) multi-head attention, 3) strided mask, and 4) a concatenation, as in Figure 3.

Strided Split & Multi-Head Attention In a regular split of a sequence of length n into c blocks, each block would be of length n/c and the first word of block $i+1$ is exactly one block, which is n/c words, apart from the first word of block i. In our strided split, we generate c blocks of length $r = n/(c-1)$, where the distance between the first word of block $i+1$ and block i is the stride s, where $s = n/(c+1)$ words. We perform this strided split on the inputted query, key, and value, generating c smaller queries, c smaller keys, and c smaller values. Each generated triplet $(Q_i, K_i, V_i), i \in [1..c]$ is then passed through a multi-head attention, which returns output A_i.

Strided Mask & Concatenation To combine these individual A_i attention outputs, we wish to simply concatenate them together. However, each A_i is of length $r > n/c$, so simply concatenating all c blocks would result in an output longer than n. As such, we must mask each A_i such that it is only of length n/c. If we naively masked the same portion of each A_i, such as the first n/c words, then we would be losing words from the original input, such as the last $r - (n/c)$ words in the last block. Instead, we must perform a strided mask, where use the first n/c words of A_1, we use the last n/c words of A_c, and we use the middle n/c words of $A_{c/2}$. We show pseudocode for the case of $c = 3$ splits of length $r = n/2$, stride $s = n/4$ below:

procedure STRIDEDSPLIT(seq, n)
    return $seq[0 : 2n/4], seq[n/4 : 3n/4], seq[2n/4 : 4n/4]$
end procedure
procedure STRIDEDMASK(seq, n, i)
    return $seq[(i \cdot n)/12 : ((i+4) \cdot n)/12]$
end procedure
procedure STRIDEDNEIGHBORHOODATTENTION(Q, K, V)
    $n \leftarrow \text{len}(Q)$
    $Q_{1,2,3}, K_{1,2,3}, V_{1,2,3} \leftarrow \text{StridedSplit}(Q, n), \text{StridedSplit}(K, n), \text{StridedSplit}(V, n)$
    $A_{1,2,3} \leftarrow \text{MultiHead}(Q_1, K_1, V_1), \text{MultiHead}(Q_2, K_2, V_2), \text{MultiHead}(Q_3, K_3, V_3)$
    $A_{1,2,3} \leftarrow \text{StridedMask}(A_1, n, 0), \text{StridedMask}(A_2, n, 1), \text{StridedMask}(A_3, n, 2)$
    return $\text{Concat}(A_1, A_2, A_3)$
end procedure

4 Experiments

4.1 Datasets

We focus on conditional language modelling, specifically single-document summarization. In our proposal and milestone, we experimented on the Wikihow dataset, but we lacked the proper baselines to compare with other state-of-the-art summarization tasks which were mostly performed on the CNN/Dailymail dataset. Thus, we preprocessed the CNN/Dailymail dataset with tokenization, while keeping tags in the target as we found that it empirically improves inference, which can be removed after the inference step. The data was split into train, validation and test sets in a 92/4/4 ratio into `src.txt` and `tgt.tagged.txt` files, where each article was written in a line in `src.txt` and the corrsponding line in `tgt.tagged.txt` represents the summarization of the article. The sequence length in the dataset ranges from 250 tokens to 16652 tokens, with a mean of 4153 and standard deviation of 2014 tokens.

4.2 Experimental Details

In our preprocessing step, we initially truncate our articles to have a max length of 400 tokens, since longer sequences give rise to memory error in Azure NV GPU machines. We first experiment with token lengths of 400, and hypothesize that convolutional and strided attention allow longer sequences to fit in memory. We also truncate the summaries' length to 100 tokens, while also having a shared

vocabulary that allows shared embeddings between the source, target and the generator by ensuring that the source and target are aligned with the same dictionary.

In the positional encoding module, we add positional encodings with sinusoidal functions to input embeddings before feeding into the encoder and decoder modules which allows the model to attend to relative positions. Each dimension i has a unique frequency and offset as shown below.

$$PE_{pos,2i} = \sin\left(\frac{pos}{1000^{2i/d_{model}}}\right), PE_{pos,2i+1} = \cos\left(\frac{pos}{1000^{2i/d_{model}}}\right)$$

During training, we applied the copy mechanism that allows the summarizer to fall back and copy from the source text when it encounters <unk> tokens by referencing to the softmax of the multiplication between attention scores of the output with the attention scores of the source. Using 4 layers in the encoder and decoder, with dimensions of $d_{model} = 512$ for multi-headed attention and $h = 8$ unique heads, we optimized over cross entropy loss between the true summary and the summarized output.

We used $p_{dropout} = 0.2$ for dropout probabilities, and Noam learning rate decay scheme (Appendix: Figure 4) with 8000 warm up steps and Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.998$ and initial learning rate of 2. Building on top of the baseline, in the memory compressed attention, we applied filters of size 3 and stride 3 and convolved the key and queries before feeding them into the multi-headed attention module. For strided attention, we followed the schema in Figure 3, and applied strided split to the queries, keys and values, while applying strided mask to make sure the dimensions of the mask match before feeding the three subsequences into the same multi-headed attention module. Thus, the size of the sequence fed into the multi-headed attention is reduced by half.

During inference time, we used beam-search with a beam size of 10, and applied a length penalty by Wu et. al. that penalizes long summarized sequences. We also prevented the model from repeating trigrams, while also applying a penalty that prevents repeated attention to the same source words that encourages better recall and attends to the whole article instead of focusing on a few sentences.

4.3 Evaluation Method

We evaluate our results during test time every 10000 steps on our validation set, and we have four main metrics: speed (as measured in tokens per second), ROUGE, perplexity and accuracy. Accuracy and perplexity are online metrics that allows us to keep track of progress in training along with cross-entropy loss, as they can be easily evaluated, especially with accuracy as a bag-of-words metric that that counts the proportion of correct words in inference as compared to the true summary.

We measure speed in training and inference time, as one of our goals of our research is to increase the efficiency of existing approaches. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a metric that measures co-occurrence statistics (18). There are numerous different flavors of ROUGE that we utilize, including ROUGE-N with $N = 2$ and ROUGE-L-F1. (16). ROUGE-N is a measure of the overlap of N-grams between the system and reference summaries, and is given by:

$$\text{ROUGE-N} = \frac{\sum_{C \in RSS} \sum_{gram_n \in C} COUNT_{match}(gram_n)}{\sum_{C \in RSS} \sum_{gram_n \in C} COUNT(gram_n)}$$

where RSS is a set of reference summaries-reference summary set, $COUNT_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and a reference summary and $Count(gram_n)$ is the number of n-grams in the reference summary.

We also utilize ROUGE-L-F1, which is a longest common subsequence measurement that takes into account sentence level structure similarity and identifies the longest co-occurring sequence n-grams. We use this type of ROUGE (especially the F1 type) as it is more appropriate in this setting, since we do not want to explicitly constrain the output length. (12) Furthermore, this metric is used commonly in text summarization, allowing us to effectively compare our results with other approaches.

5 Results

We begin by first reporting the accuracy, perplexity and speeds of baselines and our models, with additional materials such as loss curves and attention visualizations attached in the Appendix. For consistency, and to allow for a better basis of comparison, we report token processing speed, as

Table 2: Figures from epoch 10 on truncation length of 400, with experiments ran on two M60 GPUs in parallel. n denotes sequence length, s denotes stride length, and c denotes number of splits.

Attention	Accuracy		Perplexity		Speed (Tokens/s)		Theoterical Runtime
	Training	Validation	Training	Validation	Training	Inference	
Baseline	56.12	56.55	7.65	9.26	5.96	54.48	$O(n^2)$
Conv.	56.51	56.12	7.50	9.41	6.28	57.70	$O((\frac{n}{s})^2)$
Strided	**56.62**	**56.77**	**7.49**	**9.01**	**6.29**	**58.18**	$O(\frac{n^2}{c})$

Table 3: Evaluated model trained after 40k steps/10 epochs. No ensemble of models used here.

	Baseline Attention	Convolutional Attention	Strided Attention
ROUGE-1 Recall	38.60	**39.26**	37.79
ROUGE-1 Precision	41.90	41.19	**42.33**
ROUGE-1 F Score	**38.82**	38.77	38.53
ROUGE-2 Recall	16.64	**16.86**	16.40
ROUGE-2 Precision	18.47	18.06	**18.83**
ROUGE-2 F Score	16.89	16.80	**16.91**
ROUGE-L Recall	35.62	**36.38**	34.93
ROUGE-L Precision	38.76	38.27	**39.24**
ROUGE-L F Score	35.87	**35.97**	35.67

measured in tokens per second. We train our model for 40,000 steps, which is equivalent to about 10 epochs for all three models, and run inference on our model at 40k steps on the test set without model ensemble. On a source truncation length of 400 for the article to be summarized, our models performs 5.5% faster in training time, and 6.8% faster in inference time with beam search (beam size $= 10$) and copy mechanism, ceteris paribus.

In the table, we see that both our convolutional model and our neighborhood attention model achieve faster speeds than the transformers baseline, which is what we expected because in the baseline model, attention is applied over the entire document, which is the bottleneck. On the other hand, in both the neighborhood model and the convolutional model, we restrict attention size significantly, as discussed in Section 3.3. For each of our models, we also report ROUGE-1, ROUGE-2 and ROUGE-L scores as below. Better results could have been obtained with ensemble of models in past checkpoints, or by training for longer epochs (we trained for 2 days on M60 GPU for 40k steps, which is one-fifth of what Gehrmann et. al. trained for). Also by testing across different sequence lengths, we find that the strided attention has consistently higher speeds as compared to the baseline and convolutional attention during training time, with a speedup of 121% when run an sequence lengths of 600. The corresponding figures are attached in the Appendix for reference.

6 Analysis

In this section, we provide qualitative analysis of the three models and discuss the results. The most important takeaway from our previous section, where we reported our results, is that we were able to achieve fairly significant speed-ups from our baseline model in both train and inference time. This is the anticipated outcome because in our convolutional and strided attention models, we apply attention to a more restricted area. Thus, the improvements in speed match our expectations.

In terms of evaluation of the models, we see that our models achieve metrics that are on par with, if not better than, the baseline model. We display two examples of input/output sequences in Table 4 in the Appendix. In the table, we show the input, which is the first 400 words of a news article. For each article, we applied each of the three models on it, and obtained three different summarizations. From the outputs, we make several observations. Firstly, we note that our convolutional attention model is remarkably similar to the baseline model. In fact, both the models output the same exact sequence for the first example in the table.

Table 4: Table showing two example input news articles to our models. For each of the three models, we show the summarized output generated. Input articles and output generated are shortened to save space. The full table is unshortened in Appendix Table 5 and is only shown here for completion.

Document to Summarize (model input)	Baseline	Convolutional	Strided
five americans who were monitored for three weeks ... almost all the deaths have been in guinea, liberia and sierra leone. more than 10,000 ...died... ebola is spread by direct contact... .	the last of 17 ... released ... more than 10,000 people died...	the last of 17 ... released ... more than 10,000 people died...	more than 10,000 people ... in guinea, liberia and sierra leone.
a year ago bloomberg published a story with the following ... arkansas gov. asa hutchinson, and likely 14 other states considering...	mike pence has scored a lot ... rush in to defend pence and the law.	mike pence is drawing huge heat ... scored a lot of points this week among ultraconservatives.	mike pence is drawing heat for ... points this week among ultraconservatives.

While getting the same exact output sequence is not expected (and is an anomaly in our dataset as verified manually), it is not surprising that both models have very similar outputs. As discussed in Section 3.3.1, our convolutional attention model is very similar to the baseline, with the only difference being the application of a convolutional layer before in order to shorten the input sequence. Thus, it is expected that both models have similar outputs. The fact that they obtained the same exact output sequence for the first example presented in the table illustrates that our convolutional model is able to learn the same things the baseline is, albeit with a shorter sequence, indicating that using the full sequence is not only unnecessary, but computationally inefficient. It may be possible (and we leave it as future work) to obtain results just as good by shortening the input sequence even more through the convolutional layer.

Secondly, we see that the strided attention model provides noticeably different outputs than both our convolutional model and the baseline. This is also an expected outcome, as this model has the most different architecture, emphasizing short-range dependencies. Thirdly, we see that both our convolutional attention and strided attention models generate summaries that are on par, if not better, than the baseline. More specifically, our two new models generally outperform the baseline on longer input documents. This is evident in the second example in Table 5 in the Appendix, where both our convolutional and strided attention model generate subjectively better summaries of the article. The second article displayed in the table is a rather long article of 960 words. The baseline clearly struggles with this example, generating a summary that does not get the main points of the article across. The reason for this is that it is trying to attend to too much, and thus, when the input article is long (like in this example), it loses focus. On the other hand, our strided attention and convolutional models produce subjectively better summaries that encapsulate the articles main points.

In the previous section, we see that our local strided attention model is able to achieve higher precision values, while our convolutional attention model is able to achieve higher recall. The reason for this is that our convolutional model has a wider field of view, while the strided model focuses better on the local neighborhood, thus obtaining better precision.

7 Conclusion

In this paper, we presented two novel models that are design architectural improvements to transformers that allow for more efficient training while maintaining (and even exceeding) comparable metrics to existing state-of-the-art methods on document summarization. Through our analysis of the models, we saw that each model has its own unique advantage that it brings to the table, suggesting that the architectures of the models are promising. We demonstrate that both models are more computationally efficient than the current state-of-the-art, and tend to achieve better results on longer input sequences. As next steps, combining the models might result in even better performance.

Most importantly, our work shows that while transformers are the current state-of-the-art for numerous different sequence transduction tasks, there is a great deal of room for improvement in many of these tasks, both efficiency and performance wise.

References

[1] H. Bahuleyan, L. Mou, O. Vechtomova, and P. Poupart. Variational attention for sequence-to-sequence models. *CoRR*, abs/1712.08207, 2017.

[2] S. Chopra, M. Auli, and A. M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, 2016.

[3] Z. Dai*, Z. Yang*, Y. Yang, W. W. Cohen, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-XL: Language modeling with longer-term dependency, 2019.

[4] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser. Universal transformers. *CoRR*, abs/1807.03819, 2018.

[5] F. Dutil, Ç. Gülçehre, A. Trischler, and Y. Bengio. Plan, attend, generate: Planning for sequence-to-sequence models. *CoRR*, abs/1711.10462, 2017.

[6] S. Fernández, A. Graves, and J. Schmidhuber. An application of recurrent neural networks to discriminative keyword spotting. In *Proceedings of the 17th International Conference on Artificial Neural Networks*, ICANN'07, pages 220–229, Berlin, Heidelberg, 2007. Springer-Verlag.

[7] S. Gehrmann, Y. Deng, and A. M. Rush. Bottom-up abstractive summarization. *CoRR*, abs/1808.10792, 2018.

[8] A. Graves. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711, 2012.

[9] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer. Generating wikipedia by summarizing long sequences. *International Conference on Learning Representations*, 01 2018.

[10] Y. Keneshloo, T. Shi, N. Ramakrishnan, and C. K. Reddy. Deep reinforcement learning for sequence to sequence models. *CoRR*, abs/1805.09461, 2018.

[11] Z. Li, X. Jiang, L. Shang, and H. Li. Paraphrase generation with deep reinforcement learning. *CoRR*, abs/1711.00279, 2017.

[12] C.-Y. Lin and F. J. Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *ACL*, 2004.

[13] J. R. Medina and J. Kalita. Parallel attention mechanisms in neural machine translation. *CoRR*, abs/1810.12427, 2018.

[14] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2018.

[15] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685, 2015.

[16] N. Schluter. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 41–45. Association for Computational Linguistics, 2017.

[17] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy. Neural abstractive text summarization with sequence-to-sequence models. *CoRR*, abs/1812.02303, 2018.

[18] J. Steinberger and K. Jezek. Evaluation measures for text summarization. *Computing and Informatics*, 28:251–275, 01 2009.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 06 2017.

[20] K. Xu, L. Wu, Z. Wang, Y. Feng, and V. Sheinin. Graph2seq: Graph to sequence learning with attention-based neural networks. *CoRR*, abs/1804.00823, 2018.

8 Additional Information

We are doing a custom project with guidance from Kevin Clark, without external collaborators and without sharing this project for other classes.

9 Contributions

As per our milestone, we wrote the preprocessing scripts for the dataset and the validation scripts ourselves, as well as core sections of the transformer such as the self-attention module, train and test iterators, and the main train function in Pytorch. Other implementation details which weren't as familiar for us such as positional encoding, Noam's learning rate decay, and label smoothing were written with guidance from the OpenNMT's Annotated Transformer and Vaswani's paper. Moreover, we also build upon OpenNMT's implementation that includes beam search and the copy attention mechanism that prevents <unk> tokens in the output, which allows us to compare with state-of-the-art algorithms. Thus, we have our own implemented baselines and improved attention modules built upon OpenNMT, with us writing our improved attention mechanisms ourselves.

10 Appendix

In this section, we present supplementary figures, tables, and equations that we were unable to present in the main portion of the paper.

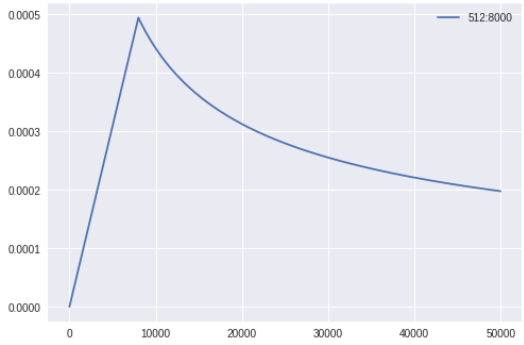

Figure 4: Learning rate scheduler with 2000 warm-up steps, and Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and initial learning rate of 1.

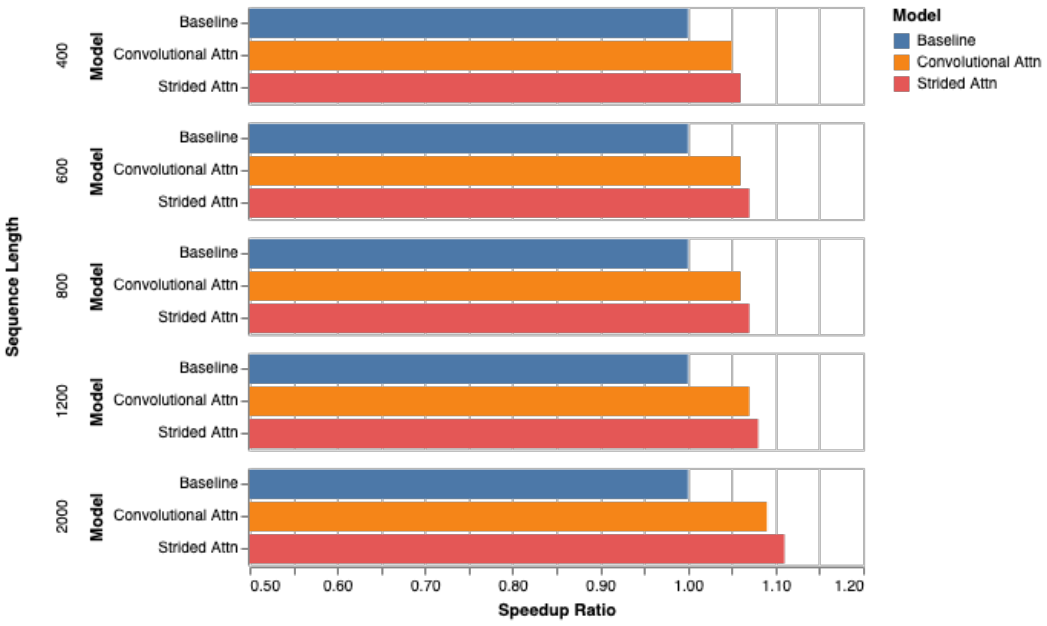

Figure 5: Speedup ratio with baseline as default (1.0) across different sequence lengths.

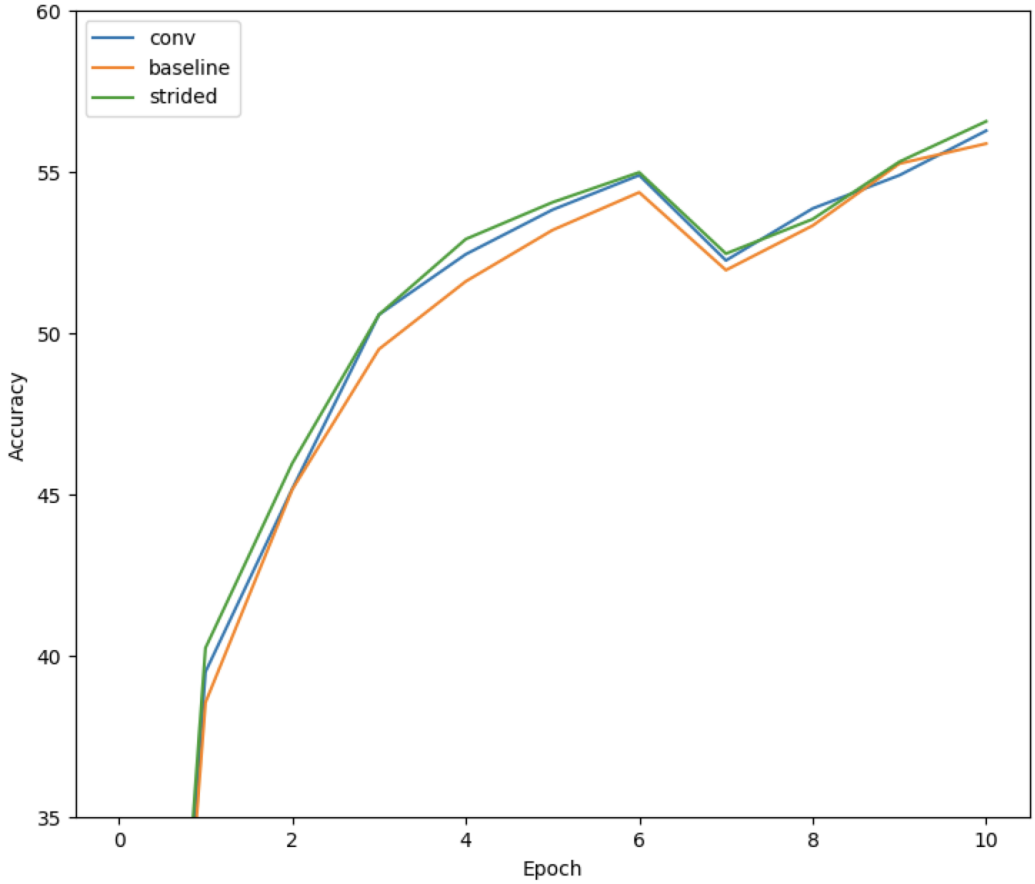

Figure 6: Accuracy curve comparing the three models during training.

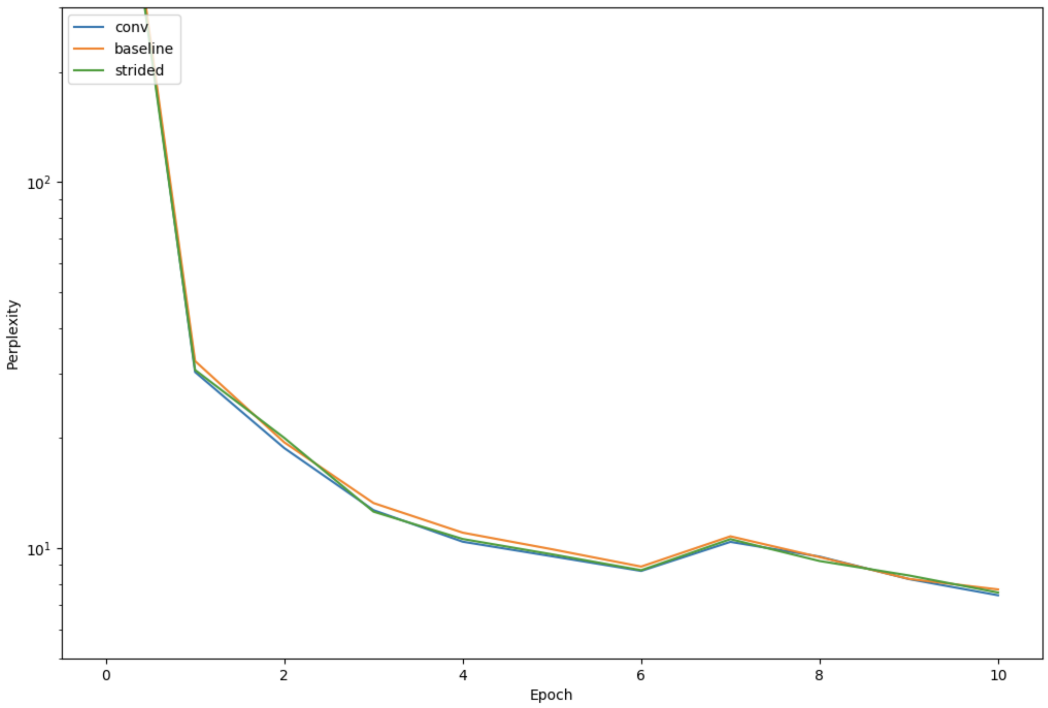

Figure 7: Perplexity curve comparing the three models during training.

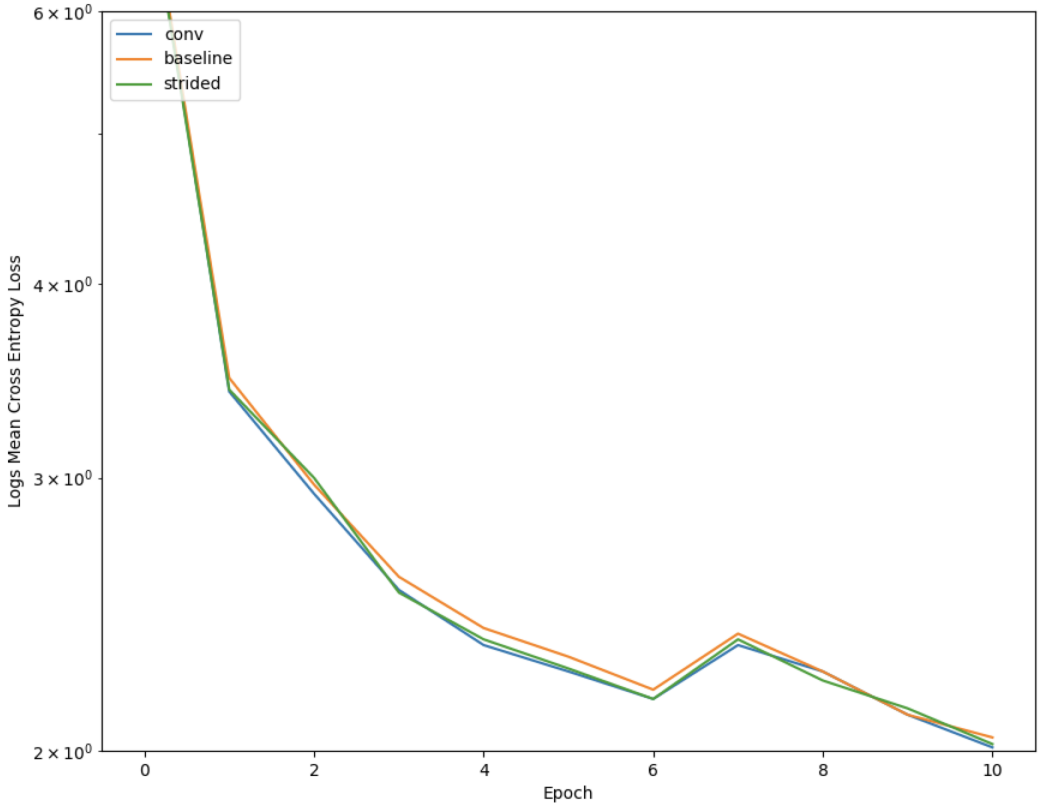

Figure 8: Loss curve comparing the three models during training.

Table 5: Examples

Document to Summarize (model input)	Baseline Attention	Convolutional Attention	Strided Attention
five americans who were monitored for three weeks at an omaha , nebraska , hospital after being exposed to ebola in west africa have been released , a nebraska medicine spokesman said in an email wednesday. one of the five had a heart-related issue on saturday and has been discharged but hasn't, left the area, taylor wilson wrote. the others have already gone home. they were exposed to ebola in sierra leone in march, but none developed the deadly virus. they are clinicians for partners in health, a boston-based aid group. they all had contact with a colleague who was diagnosed with the disease and is being treated at the national institutes of health in bethesda, maryland. as of monday, that health care worker is in fair condition. the centers for disease control and prevention in atlanta has said the last of 17 patients who were being monitored are expected to be released by thursday. more than 10,000 people have died in a west african epidemic of ebola that dates to december 2013, according to the world health organization. almost all the deaths have been in guinea, liberia and sierra leone. ebola is spread by direct contact with the bodily fluids of an infected person.	the last of 17 patients who were being monitored are expected to be released by thursday. more than 10,000 people have died in a west african epidemic of ebola that dates to december 2013.	the last of 17 patients who were being monitored are expected to be released by thursday. more than 10,000 people have died in a west african epidemic of ebola that dates to december 2013.	more than 10,000 people have died in a west african epidemic of ebola that dates to december 2013. almost all the deaths have been in guinea, liberia and sierra leone.
a year ago bloomberg published a story with the following headline: mike pence, a koch favorite, mulls 2016 run for president. the story ticked off items on pence's, conservative things-to-do list while also noting his close ties to the deep-pocketed koch brothers, as well as other right-wing lobbying groups. last august the indiana governor was in dallas for an americans for prosperity event; the group is backed by the conservative koch brothers, and supported gov. pence's, tax-slashing budget. now, pence is drawing huge heat for his controversial decision to sign a religious freedom law last week that opens the door to discrimination against gays and lesbians. why would pence ignore the pleas of indiana's, chamber of commerce as well as the republican mayor of his state capital and sign such a bill ? because there's, a very powerful wing of his party that wants a conservative as its 2016 candidate and this bill was pence's, way of shoring up his street cred. it is also the reason why republican jeb bush, pence's, fellow white house hopeful, who is viewed as a little light in that category, was first to rush in to defend pence and the law. one lesson here: just because more than 70% of the country now lives in states where same-sex marriage is legal does not mean 70% of the country is happy about it. backlash aside, the fact is pence has scored a lot of points this week among ultraconservatives. and while that may not be enough to get him over this political hump, the very public debate that now embroils him – and arkansas gov. asa hutchinson, and likely 14 other states considering...	mike pence has scored a lot of points this week among ultra-conservatives. the indiana governor was in dallas for an americans for prosperity event. pence was first to rush in to defend pence and the law.	mike pence is drawing huge heat for his controversial decision to sign a religious freedom law last week that opens the door to discrimination against gays and lesbians. because there's a very powerful wing of his party that wants a conservative as its 2016 candidate and this bill was pence's way of shoring up his street cred. pence has scored a lot of points this week among ultraconservatives.	mike pence is drawing heat for his controversial decision to sign a religious freedom law last week. the indiana governor was in dallas for an americans for prosperity event. pence has scored a lot of points this week among ultra-conservatives.