Multi-Granular Sequence Encoding via Dilated Compositional Units for Reading Comprehension

Yi Tay†, Luu Anh Tuan$^{\psi}$, Siu Cheung Hui$^{\phi}$
†`ytay017@e.ntu.edu.sg`
$^{\psi}$`at.luu@i2r.a-star.edu.sg`
$^{\phi}$`asschui@ntu.edu.sg`
$^{\dagger,\phi}$School of Computer Science and Engineering, Nanyang Technological University
$^{\psi}$A*Star, Institute for Infocomm Research, Singapore

Abstract

Sequence encoders are crucial components in many neural architectures for learning to read and comprehend. This paper presents a new compositional encoder for reading comprehension (RC). Our proposed encoder is not only aimed at being fast but also expressive. Specifically, the key novelty behind our encoder is that it explicitly models across multiple granularities using a new dilated composition mechanism. In our approach, gating functions are learned by modeling relationships and reasoning over multi-granular sequence information, enabling compositional learning that is aware of both long and short term information. We conduct experiments on three RC datasets, showing that our proposed encoder demonstrates very promising results both as a standalone encoder as well as a complementary building block. Empirical results show that simple Bi-Attentive architectures augmented with our proposed encoder not only achieves state-of-the-art / highly competitive results but is also considerably faster than other published works.

1 Introduction

Teaching machines to read, comprehend and reason lives at the heart of reading comprehension (RC) tasks (Rajpurkar et al., 2016; Lai et al., 2017; Dunn et al., 2017; Kočiskỳ et al., 2017). In these tasks, the goal is to answer questions based on a given passage, effectively testing the learner's capability to understand natural language. This has been an extremely productive area of research in the recent years, giving rise to many highly advanced neural network architectures (Xiong et al., 2016; Weissenborn et al., 2017; Seo et al., 2016; Hu et al., 2017; Shen et al., 2017; Wang and Jiang, 2016b; Wang et al., 2018). A common denominator in many of these models is the compositional encoder, i.e., usually a bidirectional recurrent-based (LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014)) encoder that sequentially parses the text sequence word-by-word. This helps to model compositionality of words, capturing rich and complex linguistic and syntactic structure in language.

While the usage of recurrent encoder is often regarded as indispensable in highly complex RC tasks, there are still several challenges and problems pertaining to its usage in modern RC tasks. Firstly, documents can be extremely long to the point where running a BiRNN model across a long document is computationally prohibitive[1]. This is aggravated since RC tasks can be easily extended to reasoning over multiple long documents. Secondly, recurrent encoders have limited access to long term context since each word is sequentially parsed. This restricts any form of multi-sentence and intra-document reasoning from happening within compositional encoder layer.

To this end, we propose a new compositional encoder that can either be used in place of standard RNN encoders or serve as a new module that is complementary to existing neural architectures. Our proposed encoder leverages *dilated compositions* to model relationships across multiple granularities. That is, for a given word in the target sequence, our encoder exploits both long-term (far) and short-term (near) information to decide how much information to retain for it. Intuitively, this can be interpreted as learning to compose based on modeling relationships between word-level, phrase-level, sentence-level, paragraph-level and so on. The output of the dilated composition mechanism acts as gating functions, which are then used to learn compositional representations of the input sequence.

[1]Many recent works tackle this issue (Yu et al., 2018; Choi et al., 2017). However, this work presents a complementary/orthogonal approach to many of these works.

A brief high-level overview to our proposed encoder is given as follows: Firstly, sequences are chunked into blocks based on user-defined (hyper-parameter) block sizes. Block sizes are often dilated in nature, i.e., $1, 2, 4, 10, 25$, etc., in order to capture more long-term information. Our encoder takes the neural bag-of-words representation of each block size and compresses/folds all words (that reside in each block) into a single summed embedding. All blocks are then passed into fully-connected layers and re-expanded/unfolded to their original sequence lengths. For each word, the gating vectors are then constructed by modeling the relationships between all blocks that this word resides in. As such, this can be interpreted as a divide-and-conquer sequence encoding method.

This has several advantages. Firstly, we enable a major speedup by avoiding either costly step-by-step gate construction while still maintaining interactions between neighboring words. As such, our model belongs to a class of architectures which is inspired by QRNNs (Bradbury et al., 2016) and SRUs (Lei and Zhang, 2017). The key difference is that our gates are not constructed by convolution layers but explicit block-based matching across multiple ranges, both long and short. Secondly, modeling at a long range (e.g., 25 or 50) enables our model to look further ahead as opposed to only one step forward. As such, the learned gates not only possess information about nearby words but also a larger overview of the context. This is in similar[2] spirit to self-attention, albeit occuring within the encoder. Thirdly, the final gates are formed by modeling relationships between multi-range projections (n-gram blocks), allowing for fine-grained intra-document relationships to be captured. The overall contributions of our work are as follows:

- We propose DCU (Dilated Compositional Units[3]), a new compositional encoder for both fast and expressive sequence encoding. We propose an overall architecture that utilizes DCU within a Bi-Attentive framework for both multiple choice and span prediction RC tasks. DCU can be used as a standalone (without RNNs) for fast reading and/or together with RNN models (i.e., DCU-LSTM) for more expressive reading.
- We conduct extensive experiments on three large-scale and challenging RC datasets - RACE (Lai et al., 2017), SearchQA (Dunn et al., 2017) and NarrativeQA (Kočiský et al., 2017). Our model is lightweight, fast and efficient, achieving state-of-the-art or highly competitive performance on three datasets.
- Despite its simplicity, our model outperforms highly complex models such as Dynamic Fusion Networks (DFN) (Xu et al., 2017) on RACE. While DFN takes approximately a week to train, spending at least several hours per epoch, our model converges in less than 12 hours with only $4-5$ minutes per epoch. Moreover, our model outperforms DFN by $2\%-6\%$ on the RACE benchmark and other strong baselines such as the Gated Attention Reader by 10%. On RACE, we outperform DFN without any recurrent and convolution layers. Ablation studies show an improvement of up to 6% when using DCU over a LSTM/GRU encoder.

[2]It is good to note that our approach explicitdly compares across blocks of multi-granularities while self-attention compares on a word-level basis.

[3]This model was originally known as MRU (Multi-Range Reasoning Units) and was published on ArXiv on March 2018.

2 Dilated Compositional Units (DCU)

In this section, we describe our proposed DCU encoder.

2.1 Dilated Composition Mechanism

The inputs to the DCU encoder are (1) a sequence $\{w_1, w_2 \cdots w_\ell\}$, and (2) list of ranges $\{r_1, r_2 \cdots r_k\}$ where k is the number of times the fold/unfold operation is executed. The final output of the encoder is a sequence of vectors which retains the same dimensionality as its inputs. Figure 1 provides an illustration of the overall encoder architecture.

2.1.1 Fold Operation

This section describes the operation for each r_j. For each r_j and the input sequence, the fold operation performs the summation of every r_j word. This is essentially the NBOW (neural bag-of-words) representation. This reduces the overall document length to ℓ/r_j where each item in the sequence is the sum of every r_j word. Given the new sequence of ℓ/r_j tokens, we then pass each token into a single layered feed-forward neural network:

$$\bar{w}_t = \sigma_r(\mathbf{W}_a(w_t)) + \mathbf{b}_a \qquad (1)$$

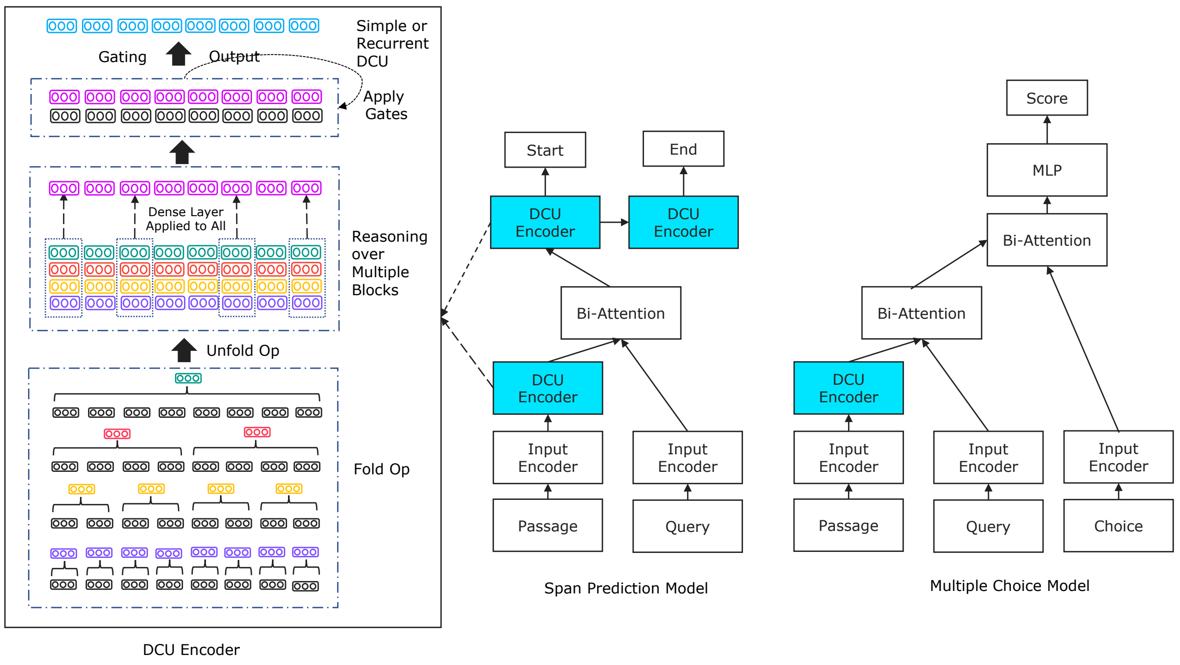

Figure 1: High-level overview of (1) our proposed DCU encoder (left), (2) Span Prediction Architecture (center) and (3) Multiple Choice Architecture (right). In the DCU encoder, blocks are formed at multi-granular levels. A block embedding is learned for each granularity. The composition gates for each word is constructed by modeling the relationships between all NBOW (neural bag-of-words) blocks that it resides in.

where $\mathbf{W}_a \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_a \in \mathbb{R}^d$ are the parameters of the fold layer. σ_r is the ReLU activation function. w_t is the t-th token in the sequence.

2.1.2 Unfold Operation

Given the transformed tokens $\bar{w}_1, \bar{w}_2 \cdots \bar{w}_{\ell/r_j}$, we then expand/unfold them into the original sequence length. Note that for each r_j, the parameters $\mathbf{W}_a, \mathbf{b}_a$ are not shared between blocks. Figure 2 depicts the fold-unfold operation for a single value of r_j.

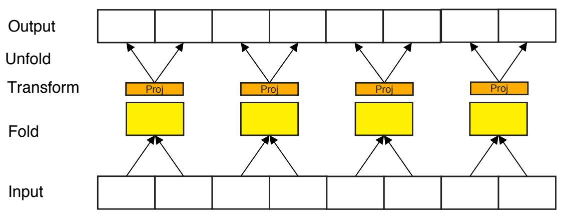

Figure 2: Overview of the Fold-Unfold operation for $r_j = 2$.

Overall, the key intuition of *each* fold-unfold operation is to learn representations of a block of a single granularity (say, blocks of 2). The main rationale for unfolding is to enable reasoning over multiple blocks (or granularities). This is described in the next section.

2.1.3 Multi-Granular Reasoning

From k different calls of the Fold/Unfold operation at different block sizes, we pass the concatenated vector of all transformed tokens into a two layered feed-forward neural network.

$$g_t = \mathbf{F}_2(\mathbf{F}_1([w_t^1; w_t^2; \cdots w_t^k])) \qquad (2)$$

where $\mathbf{F}_1(.), \mathbf{F}_2(.)$ are feed-forward networks with ReLU activations, i.e., $\sigma_r(W_x + b)$. $[;]$ is the concatenation operator. g_t is interpreted as a gating vector learned from multiple granularities and Equation (2) is learning the relationships between a token's representation at multiple hierarchies depending on the values of r_j. Notably, it is easy to see that every n pairs of words will have the same gating vector where n is the lowest value of r_j. As such, the value of the unigram, i.e., $r_j = 1$ (projection of every single token) is critical as it prevents identical gating vectors across the sequence.

2.2 Encoding Operation

To learn the DCU encoded representation of each word, we consider two variations of DCU encoders.

2.2.1 Simple Encoding

In this variation, we use g_t as a gating vector to control the fine-grained balance between the projection of each word w_t in the original input document and the original representation.

$$z_t = \tanh(\mathbf{W}_p\, w_t) + \mathbf{b}_p \tag{3}$$

$$y_t = \sigma(g_t) * w_t + (1 - \sigma(g_t))\, z_t \tag{4}$$

where $\{y_1, y_2, \cdots y_\ell\}$ is the output document representation. σ is the sigmoid function. Note that this formulation is in similar spirit to highway networks (Srivastava et al., 2015). However, since our gating function is learned via reasoning over multi-granular sequence blocks, it captures more compositionality and long range context. Note that an optional and additional projection may be applied to w_t but we found that it did not yield much empirical benefit.

2.2.2 Recurrent Encoding (DCU cell)

In the second variation, we consider a recurrent (sequential) variant. This is in similar spirit to QRNNs (Bradbury et al., 2016) and SRUs (Lei and Zhang, 2017) which reduces computation cost by pre-learning the gating vectors. The following operations describe the operations of the recurrent DCU cell for each time step t.

$$c_t = g_t \odot c_{t-1} + (1 - g_t) \odot z_t \tag{5}$$

$$h_t = o_t \odot c_t \tag{6}$$

where c_t, h_t are the cell and hidden states at time step t. g_t are the gates learned from the output of the multi-range reasoning step. o_t is an additional output gate learned via applying an affine transform on the input vector w_t, i.e., $o_t = W_o(w_t) + b_o$. Similar to RNNs, the Recurrent DCU parses the input sequence word-by-word. However, the cost is significantly reduced because we do not have expensive matrix operations that are executed in an non-parallel fashion. Finally, the outputs of the DCU encoder are a series of hidden vectors $\{h_1, h_2 \cdots h_\ell\}$ for each word in the sequence.

3 Overall Model Architectures

This section describes the overall model architectures that utilize DCU encoders. In our experiments, we focus on both multiple-choice based (RACE) and span prediction RC tasks (SearchQA, NarrativeQA). Since the core focus of this paper is our encoder, we briefly provide the high-level details[4] of our vanilla Bi-Attentive model. The Bi-Attentive models that are used in our experiments act as baselines, often being less complex than current competitive models such as BiDAF (Seo et al., 2016), AMANDA (Kundu and Ng, 2018) or DFN (Xu et al., 2017). Figure 1 (center and right side of the figure) provides a high-level illustration of these architectures.

[4]This is primarily due to the lack of space as the main focus of this work is the DCU. Source code will be released at `hhttps://github.com/vanzytay/EMNLP18_DCU`.

3.1 Multiple Choice Models

In the Multiple Choice (MCQ) model, there are three types of input sequences, namely Passage (P), Question (Q) and Answers (A_j). The output of the model (for each answer), is a score $s(P, Q, A_j) \in [0, 1]$ denoting the strength of A_j.

Input Encoding Each input sequence is passed into first a projection layer. To enhance the input word representations, we also include the standard EM (exact match) binary feature to each word. In this case, we use a three-way EM adaptation, i.e., $EM(P, Q), EM(Q, A)$ and $EM(P, A)$. The projected embeddings are then passed into a single-layered highway network.

Compositional Encoder In our experiments, we vary the encoder in this layer. Typical choices of encoders in this layer are LSTMs or GRUs. We vary this in our experiments in order to benchmark the effectiveness of our proposed DCU encoder. The output of this layer has the same dimensions as its inputs (typically the hidden states of a RNN model).

Bi-Attention Layer - This layer models the interactions between P, Q and A. Let $B(.)$ be a standard bidirectional attention that utilizes mean-pooling aggregation. The scoring function is the bilinear product of the nonlinearly transformed input, i.e., $F(x)_i^\top \mathbf{M} F(y)_i$. We first apply $B(P, Q)$ to form bi-attentive P^q, Q^p representations. Subsequently, we apply $B(P^q, A_j)$ to learn a vector representation for each answer. A temporal sum pooling is applied on the outputs of P^{qa}, A_j^p and concatenated to form $a_j^f \in \mathbb{R}^{2d}$.

Answer Selection Let $\{a_1, a_2 \cdots a_{N_a}\}$ be the inputs to this layer and N_a is the number of answer candidates. Motivated by the work in retrieval-based QA (Severyn and Moschitti, 2015), we include word overlap features to each answer candidate. This word overlap feature is in similar spirit to the EM feature. Each overlap operation between two sequences returns four features. We convert each answer vector a_j into a scalar via

$a_j^f = Softmax(\mathbf{W}_2(\sigma_r(\mathbf{W}_1([a_j]) + b_1) + b_2))$ where $\mathbf{W}_*, b_*$ and $* = \{1, 2\}$ are standard dense layer parameters.

Optimization The MCQ-based model minimizes the multi-class cross entropy where the number of classes corresponds to the number of choices.

3.2 Span Prediction Model

In the Span Prediction Model, the goal is to extract (or predict a span s, e) where $P[s : e]$ is the answer to the query. As such, the key interaction in this architecture is between P and Q. For most part, the model architecture remains similar especially for the input encoding layers and compositional encoder layer. The key difference is that we reduce the number of input sequence from three to two.

Input Encoding This follows the same design as the MCQ-based model, albeit for two sequences instead of three. Similarly, the two-way EM feature is added before passing into the highway layer.

Compositional Encoder This layer remains identical to the MCQ-based model.

Bi-Attention Layer We adopt a different bi-attention function for span prediction. More specifically, we use the 'SubMultNN' or the 'Mult' adaptation from (Wang and Jiang, 2016a) (which is tuned) and compare aligned sequences between P and Q to form P^q, the query-dependent passage representation.

Answer Pointer Layer In this layer, we pass P^q through a two layered compositional encoder (which is varied similar to the compositional encoder layer and will be further elaborated in our experiments.). The start pointer and end pointer are determined by $F(H_1), F(H_2)$ where H_1, H_2 are the hidden outputs from the first and second encoders respectively. $F(.)$ is a linear transform, projecting each hidden state to a scalar. We pass both of them into softmax functions to obtain probability distributions.

Optimization Following (Seo et al., 2016; Wang and Jiang, 2016b), we minimize the joint cross entropy loss of the start and end probability distributions. During inference, we follow (Wang and Jiang, 2016b) to find the best answer span.

4 Empirical Evaluation

In this section, we report our experimental results and comparisons against other published works.

4.1 Datasets

For our experiments, we use one challenging multiple choice MC dataset and two span-prediction MC datasets.

RACE (Reading Comprehension from Examinations) (Lai et al., 2017) is a recently proposed dataset that is constructed from real world examinations. Given a passage, there are several questions with four options each. The authors argue that RACE is more challenging compared to popular benchmarks (e.g., SQuAD (Rajpurkar et al., 2016)) as more multi-sentence and compositional reasoning are required. There are two subsets of RACE, namely RACE-M (Middle school) and RACE-H (High school). The latter is considered to be harder than the former.

SearchQA (Dunn et al., 2017) is a recent dataset that emulates a real world QA system. It involves extracting passages from search engine results and requiring models to answer questions by reasoning and reading these search snippets.

NarrativeQA (Kočiskỳ et al., 2017) is a recent benchmark proposed for story-based reading comprehension. Different from many RC datasets, the answers are handwritten by human annotators. We focus on the *summaries* setting instead of reading full stories since our model is targetted at standard RC tasks.

MCQ datasets are evaluated using the standard accuracy metric. For RACE, we train models on the entire dataset, i.e., both RACE-M and RACE-H, and evaluate them separately. For RACE, the model selection is based on each subset's respective development set. For SearchQA, we follow (Kundu and Ng, 2018; Dunn et al., 2017) which evaluates unigram exact match (EM) and n-gram F1 scores. For NarrativeQA, since the answers are human written and not constrained to a span that can be found in the passage, the evaluation metrics are BLEU-1, BLEU-4, Meteor and Rouge-L following (Kočiskỳ et al., 2017).

4.2 Competitor Methods

We describe the key competitors on each dataset.

RACE The key competitors are the Stanford Attention Reader (Stanford AR) (Chen et al., 2016), Gated Attention Reader (GA) (Dhingra et al., 2016), and Dynamic Fusion Networks (DFN) (Xu et al., 2017). GA incorporates a multi-hop attention mechanism that helps to refine the answer representations. DFN is an extremely complex model. It uses (1) BiMPM's matching functions (Wang et al., 2017c) for extensive matching between Q, P and A, (2) multi-hop reasoning powered by ReasoNet (Shen et al., 2017) and (3) employs reinforcement learning techniques for dynamic strategy selection. A leaderboard for this dataset is maintained at `http://www.qizhexie.com/data/RACE_leaderboard`. Note that the current state-of-the-art[5] (Radford et al., 2018), is a generative pre-training model trained on a large external corpus.

SearchQA The main competitor baseline is the AMANDA model proposed by (Kundu and Ng, 2018). AMANDA uses a multi-factor self-attention module, along with a question focused span prediction. AMANDA also uses BiLSTM layers for input encoding and at the span prediction layers. We also compare against the reported ASR (Kadlec et al., 2016) baselines which were reported in (Dunn et al., 2017).

NarrativeQA On this benchmark, we compare with the reported baselines in (Kočiskỳ et al., 2017). We compete on the summaries setting, in which the baselines are a context-less sequence to sequence (seq2seq) model, ASR (Kadlec et al., 2016) and BiDAF (Seo et al., 2016). As per reviewer request, we also benchmark a stronger competitor, namely R-NET (Wang et al., 2017b) on this benchmark. We use the open source implementation[6] at `https://github.com/HKUST-KnowComp/R-Net`.

4.3 Our Methods

Across our experiments, we benchmark several variants of our proposed DCU. The first is denoted as Sim-DCU which corresponds to the Simple DCU model described earlier. The model denoted by DCU (without any prefix) corresponds to the recurrent DCU model. Finally, the final variant is the DCU-LSTM which places a DCU encoder layer on top of a BiLSTM layer. We report the dimensions of the encoder as well as training time (per epoch) for each variant. The encompassing framework for DCU is the Bi-Attentive models described for MCQ-based problems and span prediction problems. Unless stated otherwise, the encoder in the pointer layer for span prediction models also uses DCU. However, for the Hybrid DCU-LSTM models, answer pointer layers use BiLSTMs. For the RACE-dataset, we additionally report scores of an ensemble of nine Sim-DCU models. This is to facilitate comparison against ensemble models of (Xu et al., 2017). We tune the dimensionality of the DCU cell within a range of $100 - 300$ in denominations of 50. The results reported are the best performing models on the held-out set.

4.4 Implementation Details

We implement all models in TensorFlow (Abadi et al., 2015). Word embeddings are initialized with $300d$ GloVe (Pennington et al., 2014) vectors and are not fine-tuned during training. Dropout rate is tuned amongst $\{0.1, 0.2, 0.3\}$ on all layers including the embedding layer. For our DCU model, we use range values of $\{1, 2, 4, 10, 25\}$. DCU encoders are only applied on the passage and not the query. We adopt the Adam optimizer (Kingma and Ba, 2014) with a learning rate of $0.0003/0.001/0.001$ for RACE/SearchQA/NarrativeQA respectively. The batch size is set to $64/256/32$ accordingly. The maximum sequence lengths are $500/200/1100$ respectively. For NarrativeQA, we use the Rouge-L score to find the best approximate answer relative to the human written answer for training the span model. All models are trained and all runtime benchmarks are based on a TitanXP GPU.

4.5 Experimental Results on RACE

Table 1 reports our results on the RACE benchmark dataset. Our proposed DCU model achieves the best result for both single models and ensemble models. We outperform highly complex models such as DFN. We also pull ahead of other recent baselines such as ElimiNet (Parikh et al., 2018) and GA by at least 5%. The best single model score from RACE-H and RACE-M alternates between Sim-DCU and DCU. Overall, there is a 6% improvement on the RACE-H dataset and

[5]This paper was not public at the time of EMNLP 2018 submission.

[6]Note that the authors of this repository state some differences with the original R-NET. However, they get similar scores on the SQuAD benchmark.

Model	RACE-M	RACE-H	RACE	Time
Sliding Window (Lai et al., 2017)	37.3	30.4	32.2	N/A
Stanford AR (Chen et al., 2016)	44.2	43.0	43.3	N/A
GA (Dhingra et al., 2016)	43.7	44.2	44.1	N/A
ElimiNet (Parikh et al., 2018)	N/A	N/A	44.5	N/A
Dynamic Fusion Network (Xu et al., 2017)	51.5	45.7	47.4	$\approx$8 hours (1 week*)
Bi-Attention (No Encoder)	50.6	44.0	44.9	3 min (9 hours)
Bi-Attention (250d GRU)	48.5	42.1	44.0	16 min (2 days)
Bi-Attention (250d LSTM)	50.3	40.9	43.6	18 min (2 days)
Bi-Attention (250d Sim-DCU)	**57.7**	$\underline{47.4}$	**50.4**	4 min (12 hours)
Bi-Attention (250d DCU)	$\underline{56.1}$	**47.5**	$\underline{50.0}$	12 min (20 hours)
GA + ElimiNet (Parikh et al., 2018)	N/A	N/A	47.2	N/A
DFN Ensemble (x9) (Xu et al., 2017)	55.6	49.4	51.2	N/A
Bi-Attention (Sim-DCU) Ensemble (x9)	**60.2**	**50.3**	**53.3**	N/A

Table 1: Comparison against other published models on RACE dataset (Lai et al., 2017). Competitor result are reported from (Lai et al., 2017; Xu et al., 2017). Best result for each category (single and ensemble) is in boldface. Last column reports estimated training time per epoch and total time for convergence. * is an estimated value that we obtain from asking the authors.

	Dev		Test		
Model	Acc	F1	Acc	F1	Time
TF-IDF max (Dunn et al., 2017)	13.0	N/A	12.7	N/A	N/A
ASR (Kadlec et al., 2016)	43.9	24.2	41.3	22.8	N/A
AMANDA (Kundu and Ng, 2018)	48.6	57.7	46.8	56.6	$\approx 8^*$ min
Bi-Attention† (No Encoder)	12.4	20.2	18.9	12.3	$\approx$17 sec
Bi-Attention† (150d BiLSTM)	40.0	51.3	38.6	49.0	$\approx$7 min
Bi-Attention† (300d LSTM)	40.3	48.7	38.2	46.4	$\approx$6 min
Bi-Attention† (300d Sim-DCU)	44.1	45.5	42.9	43.1	$\approx$25 sec
Bi-Attention† (300d DCU)	48.6	54.8	46.8	53.3	$\approx$2 min
Bi-Attention (200d Hybrid DCU-LSTM)	**50.5**	**59.9**	**49.4**	**59.5**	$\approx$7 min

Table 2: Experimental Results on SearchQA dataset. (Dunn et al., 2017). Unigram Accuracy and N-gram F1 are reported following (Kundu and Ng, 2018). All models with † use the same encoder in the answer pointer layer. * is an estimate running a replicated model with same batch size ($b = 256$) as our models.

1.8% improvement on the RACE-M dataset. Our Sim-DCU model also runs at 4 minutes per iteration, which is dramatically faster and simpler than DFN or other recurrent models. We believe that this finding highlights the importance of designing strong and fast baselines for the task at hand.

In general, we also found that the usage of a recurrent cell is not really crucial on this dataset since (1) Sim-DCU and DCU can achieve comparable performance to each other, (2) GRU and LSTM models do not have a competitive edge and (3) using no encoder can achieve comparable[7] performance to DFN. Finally, an ensemble of Sim-DCU models achieve state-of-the-art performance on the RACE dataset, achieving an overall score of 53.3%.

[7]Nevertheless, this suggests the importance of benchmarking good and strong baselines since a well-tuned baseline model can outperform DFN, a highly complicated model.

4.6 Experimental Results on SearchQA

Table 2 reports our results on the SearchQA dataset. We draw the reader's attention to the performance of the 300d DCU encoder. We achieve the same accuracy as AMANDA without using any LSTM or GRU encoder. This model runs at 2 minutes per epoch, making it 4 times more efficient than AMANDA (estimated, with identical batch size). While AMANDA also uses multi-factor self-attention, along with character enhanced representations, our simple DCU encoder used within a mere baseline bi-attentive framework comes close in performance. Finally, the hybrid combination, DCU-LSTM significantly outperforms AMANDA by 3%.

Contrary to MCQ-based datasets, we found that

Model	BLEU-1	BLEU-4	Meteor	Rouge-L	Time
Seq2Seq[†]	15.89	1.26	4.08	13.15	N/A
ASR[†] (Kadlec et al., 2016)	23.20	6.39	7.77	22.26	N/A
BiDAF[†] (Seo et al., 2016)	33.72	15.53	15.38	36.30	N/A
R-NET[ϕ] (Wang et al., 2017b)	34.90	20.30	18.00	36.70	N/A
Bi-Attention (300d LSTM)	31.18	15.34	14.42	32.95	≈1 hour
Bi-Attention (150d BiLSTM)	34.22	18.22	16.19	38.32	≈1 hour
Bi-Attention (300d Sim-DCU)	9.15	1.69	3.95	11.16	1 min
Bi-Attention (300d DCU)	33.28	16.15	15.84	36.65	18 mins
Bi-Attention (150d Hybrid DCU-LSTM)	**36.55**	19.79	17.87	**41.44**	≈1 hour

Table 3: Experimental Results on the NarrativeQA reading comprehension challenge (Kočiskỳ et al., 2017) using summaries. [†] are baselines reported by (Kočiskỳ et al., 2017). [ϕ] was obtained by running an open-source implementation of R-NET on the benchmark.

Sim-DCU model could not achieve comparable results to the recurrent DCU. We hypothesize that this is due to the need to predict spans. Nevertheless, the 300d DCU outperforms an LSTM encoder and remains competitive to a BiLSTM of similar dimensionality[8]. We also observe that LSTM and DCU are complementary. This shows that stacking a DCU encoder over standard LSTMs can give a performance boost relative to using each encoder separately.

4.7 Experimental Results on NarrativeQA

Table 3 reports our results on the NarrativeQA benchmark. First, we observe that 300d DCU can achieve comparable performance with BiDAF (Seo et al., 2016). When compared with a BiLSTM of equal output dimensions (150d), we find that our DCU model performs competitively, with less than 1% deprovement across all metrics. However, the time cost required is significantly reduced. The performance of our model is significantly better than 300d LSTM model while also being significantly faster. Here, we note that Sim-DCU does not produce reasonable results at all, which seems to be in similar vein to results on SearchQA, i.e., a recursive cell that processes word-by-word is mandatory for span prediction. However, our results show that it is not necessary to construct gates in a word-by-word fashion. Finally, DCU-LSTM significantly outperforms all models in terms of ROUGE-L, including BiDAF on this dataset. Performance improvement over the vanilla BiLSTM model ranges from $1\% - 3\%$ across all metrics, suggesting that DCU encoders are also effective as a complementary neural building block.

[8]In terms of representation and parameter size, we consider a 150d BiLSTM to be equivalent to a 300d LSTM for fair comparison.

5 Related Work

A diverse collection of MC datasets such as SQuAD (Rajpurkar et al., 2016) and CNN/DailyMail (Hermann et al., 2015) are readily available for benchmarking new deep learning models. New datasets have been recently released (Kočiskỳ et al., 2017; Joshi et al., 2017; Lai et al., 2017; Welbl et al., 2017; Dhingra et al., 2017; Trischler et al., 2016), claiming to involve a greater need for going beyond simple surface-level matching. As such, these datasets often emphasize the extent of compositional and multi-sentence reasoning required to tackle its questions.

In recent years, a wide range of innovative neural solutions have also been proposed, mainly involving bi-attention (Seo et al., 2016; Xiong et al., 2016; Cui et al., 2016) and answer pointers (Wang and Jiang, 2016b). Recent work also investigates the notion of multi-hop reasoning (Dhingra et al., 2016; Shen et al., 2017; Xu et al., 2017), reinforcement learning (Shen et al., 2017; Wang et al., 2017a; Hu et al., 2017), pretraining on auxilliary tasks (Radford et al., 2018; Peters et al., 2018; McCann et al., 2017, 2018) and self-matching / self-attention (Kundu and Ng, 2018; Wang et al., 2017b; Yu et al., 2018). While many of these works use BiLSTMs as standard building blocks, (Yu et al., 2018) proposed a RNN-less model architecture by utilizing components inspired by the Transformer architecture (Vaswani et al., 2017).

Our work is mainly concerned with designing an efficient encoder that is able to capture not only compositional information but also long-range and short-range information. More specifically, our recurrent DCU encoder takes on a similar architecture to Quasi-Recurrent Neural Networks (Bradbury et al., 2016) and Simple Recurrent Units (Lei and Zhang, 2017). In these models, gates are pre-learned and then applied. However, different from existing models such as QRNNs that use convolution layers as gates, we use block-based fold-unfold layers for learning gates. Our model also draws inspiration from dilation, in particular dilated RNNs (Chang et al., 2017) and dilated convolutions (Kalchbrenner et al., 2016), that intuitively help to model long-range dependencies. Notably, our work is orthogonal to recent advances that are targetted at speeding up the reading process. Such works include residual dilated convolutions (Wu et al., 2017), self-attention (Yu et al., 2018) and coarse-to-fine grained paradigm (Choi et al., 2017). However, while speed is one of the clear benefits of this work, our work is the first to introduce the idea of block-based multi-granular reasoning. We believe that this new building block is complementary/useful to the RC task in general.

6 Conclusion and Future Work

We proposed a novel neural architecture, the DCU encoder and an overall bi-attentive model for both MCQ-based and span prediction MC tasks. We apply it to three MC datasets and achieve competitive performance on all without the use of recurrent and convolution layers. Our proposed method outperforms DFN, an extremely complex model, without using any LSTM/GRU layer. We also remain competitive to AMANDA and BiDAF without any LSTM/GRU. While our proposed encoder demonstrates promise on reasoning and understanding natural language, we believe that our encoder is generalizable to other domains beyond reading comprehension. However, we defer this prospect to future work.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. http://tensorflow.org/.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks. *CoRR* abs/1611.01576.

Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang. 2017. Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*. pages 76–86.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858* .

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 209–220.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423* .

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549* .

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904* .

Matthew Dunn, Levent Sagun, Mike Higgins, Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179* .

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Mnemonic reader for machine comprehension. *arXiv preprint arXiv:1705.02798* .

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551* .

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547* .

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* .

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *arXiv preprint arXiv:1712.07040* .

Souvik Kundu and Hwee Tou Ng. 2018. A question-focused multi-factor attention network for question answering .

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683* .

Tao Lei and Yu Zhang. 2017. Training rnns as fast as cnns. *arXiv preprint arXiv:1709.02755* .

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*. pages 6297–6308.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730* .

Soham Parikh, Ananya Sai, Preksha Nema, and Mitesh M Khapra. 2018. Eliminet: A model for eliminating options for reading comprehension with multiple choice questions. https://openreview.net/forum?id=B1bgpzZAZ.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* .

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training .

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* .

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*. pages 373–382.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pages 1047–1055.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR* abs/1505.00387.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830* .

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. pages 6000–6010.

Shuohang Wang and Jing Jiang. 2016a. A compare-aggregate model for matching text sequences. *CoRR* abs/1611.01747.

Shuohang Wang and Jing Jiang. 2016b. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905* .

Shuohang Wang, Mo Yu, Shiyu Chang, and Jing Jiang. 2018. A co-matching model for multi-choice reading comprehension. *arXiv preprint arXiv:1806.04068* .

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2017a. R3: Reinforced reader-ranker for open-domain question answering. *arXiv preprint arXiv:1709.00023* .

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017b. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 189–198.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017c. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. pages 4144–4150.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural qa as simple as possible but not simpler. *arXiv preprint arXiv:1703.04816* .

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2017. Constructing datasets for multi-hop reading comprehension across documents. *arXiv preprint arXiv:1710.06481* .

Felix Wu, Ni Lao, John Blitzer, Guandao Yang, and Kilian Weinberger. 2017. Fast reading comprehension with convnets. *arXiv preprint arXiv:1711.04352* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR* abs/1611.01604.

Yichong Xu, Jingjing Liu, Jianfeng Gao, Yelong Shen, and Xiaodong Liu. 2017. Towards human-level machine reading comprehension: Reasoning and inference with multiple strategies. *arXiv preprint arXiv:1711.04964* .

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations*. https://openreview.net/forum?id=B14TlG-RW.