Lightening the Load: DeLighT Blocks for Faster QA Training and Ensembling

Stanford CS224N Default Project

Sam Lowe
Department of Computer Science
Stanford University
samlowe@stanford.edu

Abstract

Transformer models have proven to be one of the most dominant paradigms for NLP in recent years across a wide variety of problem domains. However, these networks tend to be extremely memory- and computationally-demanding models to train, leaving the door open to more efficient alternatives to standard transformer architectures. In this work, we explore the applicability of one such lightweight transformer alternative - the DeLighT block - to the task of SQuAD 2.0 question answering. The core architectural innovation of the DeLighT block is in its namesake DeLighT transform, the first component in the block, which is a series of group-linear, expand-reduce layers, with special input mixing and feature shuffling connections to allow for strong signal flow and long-term dependency learning. The remainder of the block architecture is similar to standard Transformer blocks, with self-attention and modeling layers, but with the flexibility to operate over compressed representations due to the shift in representational power towards the DeLighT transform. In our experiments, we replace the Transformer-style RNN Encoder blocks in a standard QANet model with DeLighT blocks. This substitution provides two main advantages - greater flexibility over per-block parameter counts and parallelization during training time - which results in cheaper training costs in regards to both memory and compute time. We experiment with several different model configurations of varying architectures, attention styles, and training regimes to discover a lightweight model to use as the basis for an ensemble method, in which we further experiment with a variety of DeLighT configurations before summing over their normalized predictions to determine the final answer. Our ensemble method achieves modest performance, but we demonstrate the promise of the approach by achieving near performance parity with the baseline model with one of our models at comparative parameters counts, supporting the promise of the DeLighT approach with further investigation into the correct configuration.

1 Key Information to include

- Mentor: Allan Zhou
- External Collaborators: None
- Sharing project: No

2 Introduction

Natural language processing is an exceedingly broad area of research that covers a wide range of end-user tasks - machine translation, summarization, and sentiment analysis are just one small sample. One of the core skills needed by systems to tackle any number of these tasks is that of reading comprehension: the ability to understand the information contained in a text, remember it in some

way, and then reason with it for some downstream function. The need to imbue our models with these abilities and to measure their performance on the comprehension subproblem motivates the task of question answering as a valuable area of research. In a question answering task, a system is typically presented with some piece of text to serve as the context for a question that the system is then expected to answer, based on the information it has been given. In this work, we utilize SQuAD 2.0 question answering as our problem setting [1]. For the SQuAD 2.0 dataset, the answers all take the form of spans from the text, that is, the answer is a continuous stretch of text embedded directly in the context. However, the new 2.0 version of SQuAD represents a more challenging problem, as roughly half of the questions are unanswerable (no span in the context answers the query), requiring systems to reason both deeply and confidently about the text.

Since the introduction of the architecture in the seminal paper "Attention is All You Need", Transformer models have been used to achieve state-of-the-art performance on almost as many tasks as there are problem areas in NLP [2]. In fact, the language models responsible for many of the flashiest headlines and most impressive performances in recent years have mainly been powered by Transformer architectures, with GPT-3 being one prominent example [3]. However, as exemplified by GPT-3, this increase in performance has come at a significant cost in terms of compute power and training time, with the GPT-3 model containing on the order of 175 billion parameters. This size of model places the current state-of-the-art in NLP far outside the reach of hobbyists or even many researchers, motivating the necessity of discovering smaller models, capable of competitive performance when compared to more fully-parameterized networks, in order to promote the democratization of NLP research.

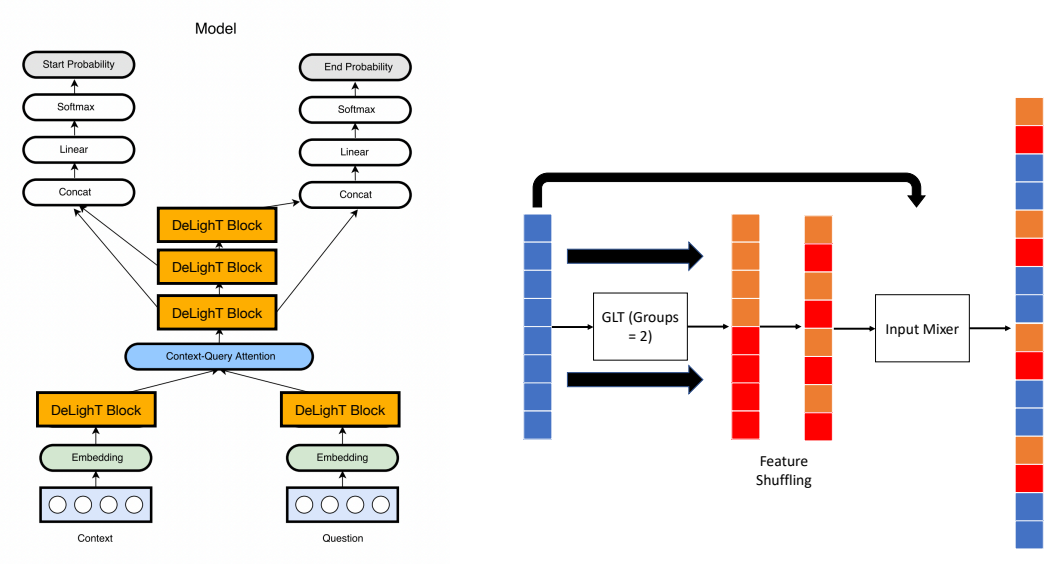
In this work, we explore the applicability of one particular lightweight Transformer alternative, the DeLighT block, to deliver smaller model sizes that still perform effectively on the SQuAD 2.0 benchmark. Although unable to exceed the baseline performance, we did uncover one model configuration that results in performance parity with the baseline at comparative parameter counts, and the drastically reduced training times proved to be a boon in the promotion of research efforts.

3 Related Work

3.1 Making NLP Models Lighter: DeLighT and DeFINE

Over the past few years, Mehta and his research collaborators have studied a variety of approaches to making models for natural language processing more lightweight and efficient. In their first paper from 2020, they present the DeFINE operation, a deep, factorized input token embedding approach for sequence models [4]. In seeking to reduce the parameter density in the lowest layers of the network (which represent a large percentage of the total parameter count for models with large vocabularies), they introduced an approach to input embedding based on group linear transforms called a hierarchical group transformation (HGT). The DeFINE architecture contains many ideas embodied in their later work on transformers, including the use of expand-reduce layers to enable downstream compression, group linear operations, and the use of input mixing connections to maximize information flow. However, the DeFINE architecture addresses a fundamentally different problem from the one under study here: transformer substitutes that address the compute bottleneck represented by their large parameter counts and, when RNN-based, their linear sequence dependencies.

In their follow-up paper "Deep and Lightweight Transformers", Mehta et al. tackled the problem of efficient transformer architectures directly by introducing a new block architecture called DeLighT based on their previous work with DeFINE [5]. The core component of the architecture is an expand-reduce operation in the style of DeFINE called the DeLighT transformation. In comparison to the DeFINE operator, DeLighT similarly utilizes group linear transformations with input mixing connections, but uses a sequence of group counts that increase over the expand layers and then decrease back to a singular group during the reduce phase, rather than in hierarchical (decreasing) order. Additionally, the DeLighT architecture uses feature shuffling before the input mixing layer in order to promote the discovery of long-range dependencies. The representational power afforded by DeLighT allows for further efficiency gains in the rest of the architecture, as the output of DeLighT can be compressed before being passed to a self-attention layer and the final modeling layers can be replaced by a reduce-expand pair. One of the core innovations of the DeLighT architecture that enables its lightweight functionality is the increased flexibility it offers when compared to standard transformers, as the layer counts and sizes in the DeLighT transformation can be independently



(a) DeLiQA, a QANet model with DeLighT blocks in place of RNNEncoders. Original figure from [6].

(b) Sample layer of DeLighT transformation.

Figure 1: Key Components of Our Approach: DeLiQA and DeLighT

adjusted, allowing for the construction of wide and deep networks when desired. The authors then explore the efficacy of their approach by demonstrating competitive performance at 50% parameter counts when compared to the state-of-the-art on a variety of task, and parity or better when expanding the DeLighT blocks to equivalent parameter counts. Despite this analysis, they did not study the performance of the DeLighT architecture in the context of modular, complex models like those used for question answering, a novel contribution of this work.

3.2 Transformers for Question Answering: QANet

As mentioned in the introduction, transformer architectures have made their way into almost every research area in NLP, and question answering is no exception. However, due to the particularities of the QA problem (like the need to deal with the question and answer in both isolation and combination), question answering systems tend to be highly complex and modular, with particular modules dedicated to specific tasks, so the approach to applying transformers to the problem is necessarily also a modular one, as is the case for the main transformer-based question answering architecture studied in this work: QANet [6]. Compared to a more general, transformer-based language model like GPT-3, which consists of a singular architecture component repeated 96 times, QANet only utilizes transformers in a limited capacity. Specifically, QANet contains transformer-style RNNEncoder blocks in the encoder layer directly before the context-to-query attention layers and in the modeling layers directly after. This architecture previously achieved state-of-the-art on the original SQuAD benchmark, and due to its performance, its use of transformers for question answering, and the need to address the peculiarities of question answering in this work, QANet will form the basis for my DeLighT substitution.

4 Approach

4.1 Baseline

Since we are working on the IID SQuAD default project track, the baseline was provided with the starter code, and is essentially a Bi-Directional Attention Flow model without the character embedding layer. We refer you to the original BiDAF paper for further details [7].

4.2 DeLiQA

We have elected to give the main approach for this work the name DeLiQA to represent its lineage as a combination of the QANet and DeLighT methods. The architecture is depicted in Figure 1(b) and is a simple alteration of the core QANet model that replaces the transformer encoders with DeLighT blocks (described in detail in the following section). More specifically, the architecture consists of an embedding layer followed by a DeLighT block that operate on the context and query inputs in isolation. The output of the first DeLighT layers is then passed to a cross-attention layer that allows the context to attend to the query and vice versa. The output of the cross-attention layer is then operated over by a series of three DeLighT blocks, though weights are shared between all three, with the output of the first two being concatenated and passed to a linear layer followed by softmax to estimate the start of the span, and the output of the latter two similarly concatenated and transformed to generate end probabilities. Given the time constraints of the project and the focus on the DeLighT block component in particular, we utilized a pre-existing implementation of QANet to bootstrap our experiments, credit to https://github.com/PanXiebit/QANet-pytorch/.

4.3 DeLighT Block

We will now more fully specify the DeLighT block, and its core DeLighT transformation, that we gave a brief overview of prior. Formally, a DeLighT transformation is specified by five parameters: the number of layers N, the maximum width multiplier w_m, the input dimension d_m, the output dimension d_o, and the maximum number of groups g_{max}. Over the first $\lceil \frac{N}{2} \rceil$ layers, the input is expanded up to a maximum dimension of $w_m d_m$, and then reduced to d_o-dimensional output over the remaining $\lfloor \frac{N}{2} \rfloor$ layers. At a given layer, the GLT operation splits the input into a number of groups, each of which is transformed with its own independent weights and biases and concatenated to form the layer output. The number of groups at a given layer increases by a factor of two over each of the first $\lceil \frac{N}{2} \rceil$ layers (capped at g_{max}) and then proceeds in reverse back to a singular group at the output. Additionally, for each layer after the first, the original input is split into the correct number of groups and spliced into a shuffled version of the previous layer's output groups. By combining feature shuffling with local weight matrices, the DeLighT transform is able to model both long- and short- term dependencies. The function of a DeLighT transform layer can be compactly specified as:

$$\mathbf{Y}^l = \begin{cases} \mathcal{F}(\mathbf{X}, \mathbf{W}^l, \mathbf{b}, g^l), & l = 1 \\ \mathcal{F}(\mathcal{H}(\mathbf{X}, \mathbf{Y}^{l-1}), \mathbf{W}^l, \mathbf{b}, g^l), & \text{otherwise} \end{cases}$$

where $\mathcal{F}$ is the group linear transform and $\mathcal{H}$ is the feature-shuffling, input-mixing operation. A sample layer of this operation is depicted in Figure 1(a).

In the full DeLighT block, this transformation is followed by a causal self-attention layer, a feedforward layer to recover the original input size d_m to enable a residual connection, a reduce-expand modeling layer pair, and finally a dropout layer and final residual addition. As the DeLighT block represents the main object of study in this work, it was coded entirely from scratch.

4.4 Ensembling

Once we established a DeLiQA architecture capable of learning (not just guessing no answer for all questions), we experimented with a variety of hyperparameter settings for the architecture in general and DeLighT block in particular. In order to maximize performance with these lighter-weight models, we performed ensembling experiments over a varying number of trained DeLiQA models. Our ensembling technique was quite simple - we simply added the normalized softmax probabilities (to remove the possibility of a single model learning higher weights and thus dominating) and re-normalized for the final prediction.

5 Experiments

5.1 Data

The dataset for the project is provided by the SQuAD 2.0 benchmark, and consists of 129,941 training samples, 6078 dev samples, and 5915 test samples [1]. As mentioned previously, each sample consists

of a question, a context paragraph, and answer (span) from the paragraph (if there is one). The goal for the model is then to predict the correct start and end token for the answer.

5.2 Evaluation method

The evaluation metrics for the SQuAD 2.0 benchmark consist of two measures: Exact Match (EM) and F1. Exact Match is a binary measure of whether the system output exactly matches the ground truth, while F1 is a more flexible measure that is the harmonic mean of precision and recall. The Exact Match and F1 scores on a particular sample are defined as the maximum score achieved across the available reference translations.

5.3 Experimental details

We performed a large number of experiments with differing architectures, training regimes, and hyperparameter settings in order to discover a configuration that would perform satisfactorily with the DeLighT block as its core encoding or modeling module. All models were trained for 30 epochs.

Baseline Modifications Our initial experiments consisted of a direct substitution of the DeLighT block for the RNNEncoder blocks in the BiDAF baseline, but we found that this substitution was unable to learn beyond the local minimum of guessing no answer. Given the difference in training regimes for QANet and BiDAF and that the former was actually transformer-based, we next elected to train the DeLighT-BiDAF model with a training regime more closely resembling QANet (Adam with warm-up and a learning rate of 0.0003), but still found the model struggled to learn anything. For both of these models, we used one DeLighT block before the cross-attention layer and two after. The DeLighT block parameters were $N = 5$, $d_m = 128$, $d_o = 64$, $w_m = 8$, and $g_{max} = 4$. Additionally, the attention layers were single-headed and the expand-reduce modeling layers had hidden size $d_m/4$.

QANet Attention Experiments Once we established our QANet implementation was working, we initially experimented with differing attention styles in our DeLighT block in the spirit of Assignment 4: causal self-attention and synthesizer attention. The DeLighT block settings were identical to the BiDAF version besides $d_o = 128$ and an increase to 4 attention heads. From a module level, we utilized three DeLighT blocks per modeling layer after cross attention (meaning the final output has been transformed by 9 DeLighT blocks, though again with shared weights). The training regime was also identical to the final BiDAF experiment (Adam with warm-up and a learning rate of 0.0003).

Varying DeLighT Block Depth + Width After observing the most promising results with causal self-attention (a model we'll call DeLiQA.1), our next experiments attempted to utilize the added flexibility of the DeLighT block to discover the strongest-performing single model. Given that we had to reduce the number of DeLighT blocks as compared to Encoder layers in the original QANet implementation due to the multiplicative effect of the GLT layer count, so we trained two alternatives to our first causal attention model. The first alternative (DeLiQA.2) had wider DeLighT transformations in the modeling layers (as a factor of input size the hidden sizes for the GLT layers were 4, 8, 8, and 4) but fewer blocks (2). In the second alternative (DeLiQA.3), the modeling DeLighT blocks had narrower transformations ($w_m = 4$) and single-headed attention but a wider reduce-expand pair ($d_m * 2$ by $d_m/2$) and more total blocks (5). Additionally, DeLiQA.3 was trained with a learning rate of 0.001 for the first 15 epochs and 0.0003 for the final 15 due to gradient collapse around the halfway point.

Ensembling Our final series of experiments utilized the ensembling technique outlined in Section 4.4 in an attempt to leverage what was learned across our 3 successful models, DeLiQA.1, DeLiQA.2, and DeLiQA.3 (DeLiQA.E).

5.4 Results

Model	Dev F1	Dev EM	Test F1	Test EM
BiDAF (Baseline)	60.77	57.47	–	–
BiDAF + DeLighT	53.42	48.21	–	–
BiDAF + DeLighT (QANet Regime)	52.70	46.77	–	–
DeLiQA (Synth Attention)	55.76	51.94	–	–
DeLiQA.1	58.10	53.97		–
DeLiQA.2	56.81	52.58	–	–
DeLiQA.3	56.81	53.12	–	–
DeLiQA.E	56.70	52.80	53.18	49.01

Table 1: Results for the experiments described in Section 5.3.

Model	Parameter Count (Millions)
BiDAF (Baseline)	2.34
DeLiQA.1	2.57
DeLiQA.2	3.13
DeLiQA.3	3.16
DeLiQA.E	8.86

Table 2: Parameter count comparisons for the DeLiQA models and QANet.

Unfortunately, our qualitative results did not meet our expectations for performance, though there are certainly bright spots that motivate further work in the area. The primary achievement of this research so far is DeLiQA.1, which achieves near performance-parity with the baseline at similar parameter counts. Due to the time I spent in searching for performant configurations, I was unable to focus as much as I would like on reducing parameter counts, though it is possible based on the results above that I ultimately overparameterized the final iterations. I was hopeful that the ensemble method would achieve the best performance of any approach, but it appears based on Table 1 that the models did not learn complementary features, as performance actually degraded. In total, the DeLighT approach appears to have promise, but the added flexibility of the various DeLighT block parameters actually makes it more difficult to find the correct combination of hyperparameters when experimenting.

6 Analysis

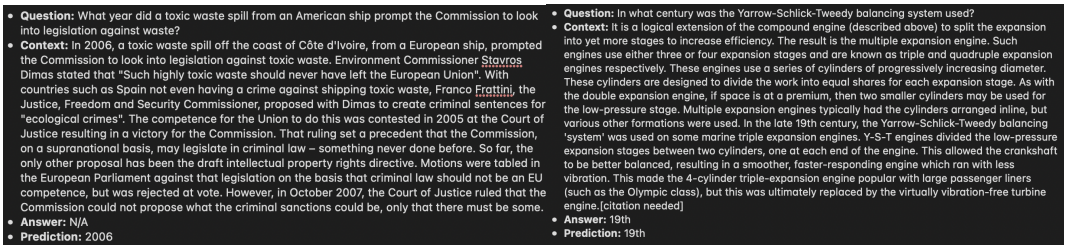
To better understand where the models performed well and where they failed, we will now look at a selection of sample outputs from our experiments.

6.1 Model Specialization

One of the more interesting observations I made when observing sample output from the models at training time is that some of the DeLighT models seemed to have a particular affinity for certain types of answers. For example, DeLiQA.1 excelled at numerical answers, particularly those involving dates, and was frequently so confident in its date-answering ability that it would predict date ranges for unanswerable questions, which is a more complex behavior to learn than the right answer in that context. Figure 2 contains representative samples.

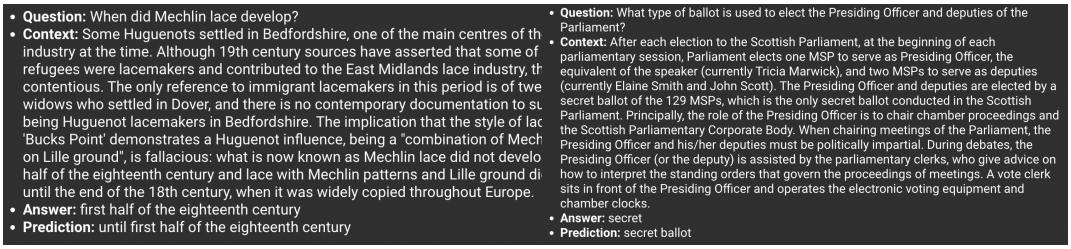
6.2 Ensembling and Confidence Diffusion

While the above analysis initially led me to believe that I would find a good deal of success in ensembling, one of the issues that ultimately resulted in the degraded performance is a phenonemon I observed in the sample outputs I have termed "confidence diffusion". While the individual models were capable of making compact span predictions, the behavior that resulted when summing their normalized outputs was widened answer spans as the ensemble attempted to interpolate between the individual model predictions, leading to answers that often contained extra words. Figure 3 contains samples of these overly-long span predictions



(a) Sample incorrect date prediction from DeLiQA.1 (b) Sample correct date prediction from DeLiQA.1

Figure 2: Date predictions from DeLiQA.1



(a) Sample incorrect date prediction from DeLiQA.1 (b) Sample correct date prediction from DeLiQA.1

Figure 3: Confidence diffusion from DeLiQA.E

7 Conclusion

In this work, we studied the applicability of the DeLighT block architecture to a wide variety of question answering contexts and training regimes. While individual model performance never exceeded the baseline, we were able to achieve near performance-parity at a comparable parameter count, supporting the claim that the DeLighT architecture can serve as a substitute for transformers in modular QA systems. Additionally, we were able to perform a large number of experiments in a relatively short window due to the gains in training speed enabled by the removal of linear sequence dependencies for RNN-based encoders. Although our ensemble performance was modest, further work on identifying the best configuration for single-model architectures could drastically improve their combined metrics. At the outset of this work, I held the belief that the increased flexibilty afforded by the granular control of the DeLighT block would be an asset in building performant architectures, but it turned out to be the largest diversion for this project, as most of my research time was spent on various tweaks to the DeLiQA model. One of the limitations of our current work is the use of the DeLighT block in contexts previously purpose-built and optimized for transformers or RNNs, meaning that research into a question answering architecture built for the DeLighT block could lead to large jumps in performance, representing a promising avenue for future development.

References

[1] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Sachin Mehta, Rik Koncel-Kedziorski, Mohammad Rastegari, and Hannaneh Hajishirzi. Define: Deep factorized input token embeddings for neural sequence modeling. *arXiv preprint arXiv:1911.12385*, 2019.

[5] Sachin Mehta1, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Delight: Deep and light-weight transformer. In *International Conference on Learning Representations (ICLR)*, 2021.

[6] Adams Wei Yu, David Dohan, and Minh-Thang Luong. Qanet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations (ICLR)*, 2018.

[7] Minjoon Seol, Aniruddha Kembhavi, and Hananneh Hajishirzi. Bi-directional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR)*, 2017.