Proceedings of the Sixth International Conference on Computing Methodologies and Communication (ICCMC 2022)
IEEE Xplore Part Number: CFP22K25-ART; ISBN: 978-1-6654-1028-1

2022 6th International Conference on Computing Methodologies and Communication (ICCMC) | 978-1-6654-1028-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCMC53470.2022.9753760

Natural Language Processing based Automated Essay Scoring with Parameter-Efficient Transformer Approach

Angad Sethi
Machine Learning Research Lab
Dept. of Computer Science
Delhi Technological University
Delhi, India
angadsethi_2k18co066@dtu.ac.in

Kavinder Singh
Machine Learning Research Lab
Dept. of Computer Science
Delhi Technological University
Delhi, India
kavinder85@gmail.com

***Abstract*—Existing automated scoring models implement layers of traditional recurrent neural networks to achieve reasonable performance. However, the models provide limited performance due to the limited capacity to encode long-term dependencies. The paper proposed a novel architecture incorporating pioneering language models of the natural language processing community. We leverage pre-trained language models and integrate it with adapter modules, which use a bottle-neck architecture to reduce the number of trainable parameters while delivering excellent performance. We also propose a model by re-purposing the bi-directional attention flow model to detect adversarial essays. The model we put forward achieves state-of-the-art performance on most essay prompts in the Automated Student Assessment Prize data set. We outline the previous methods employed to attempt this task, and show how our model outperforms them.**

***Index Terms*—Natural Language Processing, Transformers, Language Models, Automated Essay Scoring, LSTM, GRU**

I. Introduction

There has been a surging need for automated essay scoring with the emergence of essay-type questions in large-scale standardized testing. Not only this, but automated essay scoring has also surfaced as an evaluation method to test the comprehension of different models. Given its enormous scope and high levels of variability, automated essay scoring has proven ideal in testing the robustness of deep-learning algorithms and models. Chodorow & Burstein [1] attempted to solve the problem of automated essay scoring. The earliest algorithm is e-rater99 [1], which computes essay scores based solely on the essay length. The e-rater 99 was improved upon by the e-rater01, another system developed by ETS, which decreased its predecessor's reliance on essay length by considering proximity to the given topic. Klebanov *et al.* [2] introduced a significant breakthrough in the designs of automated essay scoring algorithms. Klebanov *et al.* introduced the concept of argumentation, which is an essential metric in essay scoring. It also reduced reliance on hand-engineered features such as grammatical errors and ambiguities.

The deep-learning breakthrough massively accelerated the development of models for various tasks [3]–[8]. In literature, a majority of the models built implemented recurrent neural networks, and particular flavors of them, such as LSTM cells [9], and GRUs [10]. The aforementioned models, particularly recurrence-based models (RNN), suffer from a few fundamental flaws, both in concept and implementation. Firstly, RNN based models are computationally expensive and can be parallelised after great efforts and by using sophisticated hardware [1]. The main reason for this is that RNNs cannot compute parallelly; the computation must be sequential. Due to the great length of essays which can sometimes include thousands of timesteps, LSTMs proved to be cumbersome and would take up vast amounts of computational time, even when accelerated with GPUs. Acknowledging the supremacy of deep neural networks in modelling mappings between the input and output [11], it is virtually impossible to build a stacked LSTM model which performs reasonably. Secondly, there was a problem with the word embeddings that were the heart of such models. The aforementioned word-embeddings were stagnant and did not mould according to the contexts, severely limiting their use and hampered models where essays contained words that did not adhere to their use in conventional linguistics. The advent of transformers solved the first problem [12] which stacked multiple encoder and decoder blocks and achieved SOTA performance on benchmarks such as GLUE [13] and SuperGLUE [14] and is further discussed in section II. The second problem was solved by ELMO [15], with its fine-tuning module modulated word embedding vectors according to the context. Google's BERT [16] combined these solutions to create a model that has since become the 'defacto' model in natural language processing and semantic systems, and is further discussed in section II.

Automated essay scoring models found in recent literature involve only LSTMs and CNNs. All the LSTM-based models mentioned in Section II take at least three days of GPU training time before they can achieve reasonable performance. On the other hand, CNN's cannot be used on their own because even though they select features based on importance, they lead to loss of information and need to be coupled with an LSTM based model that comes with its own set of problems. As detailed in table II, LSTM based models can not achieve more than **0.72** when averaged over all prompts.

We propose a distinct model to combat the aforementioned

[1] https://www.telesens.co/2018/07/26/understanding-roofline-charts/

problems, encompassing two components, namely a prompt-aware and a prompt-unaware component. The prompt-unaware phase implements BERT augmented by adapters [17]. Instead of fine-tuning millions of parameters, we freeze a part of the model and train only a few thousand parameters to achieve excellent performance. In the prompt-unaware phase, we are able to achieve quadratic weighted kappa (QWK) of **0.785**. For the prompt dependent phase, we adapt bi-directional Attention Flow (BiDAF) [18], a question-answering model for automated essay scoring purposes, and are able to achieve a QWK of **0.746**. We also vary the hyperparameters, including the learning rates, sequence lengths, and tokenization methods, and show the increased robustness of our model.

Our major contributions in the development of the proposed model systems are:

- Perused previous literature, handpicking best performing models.
- Developed a model with both prompt-unaware and prompt-aware components.
- Compared our model's performance against previously developed models.
- Incorporated modern encoders, like BERT and adapters, and modern concepts, like attention, into our models.
- Performed ablation studies to prove the efficacy of our models.

II. Related Works

When the computer science domain encountered the problem of automated scoring, there were multiple attempts at tackling it by using regression, i.e. treating feature values as independent variables and essay scores as dependent variables. Vapnik *et al.* [19], devised a system that implemented support vector regression to predict essay scores.

The first large-scale deployment of automated essay scoring systems began with the e-rater99 [1] which computed essay scores based solely on the essay length. The e-rater99 was a very rudimentary model and did not generalize well. However, it was essential because it sowed the seed for future models. The e-rater99 was improved upon by the e-rater01 [1], which decreased its predecessor's reliance on essay length. The model [1] depends on only hand-engineered features, including part-of-speech tagging, morphological dependencies, and requires the combined knowledge of multiple linguists. After considering features, the machine learning model outputs the final score by using a regression model [20]. However, the model requires larger computation and heavily relies on human experts; it did not scale well.

Moving into the machine learning era, Hewlett-Packard sponsored a competition on Kaggle [2], which invited entries that are capable of predicting scores on essays via machine learning [21]–[24]. A host of systems were submitted, with the best one achieving a QWK of **0.81**. This competition also provided a high-quality dataset, which has been used to prove performance by several automated essay scoring systems and is further described in section III-A.

With the advent of the deep-learning era, there has been much work in the field of essay scoring by using deep bi-directional LSTM [9] networks. Taghipour and Ng [25] devised a system that involved both convolutional networks and recurrent networks. Convolutional networks particularly help their case by convolving over the entire sequence and producing task-appropriate representations. Their best model achieves a QWK of **0.746** averaged over the prompts, and a QWK of **0.987** when computed without prompts. Nadeem *et al.* [26] implement two models: Hierarchical recurrent network with attention [27], and Bidirectional context with attention [28]. Both models employ LSTMs. HAN utilises a word $\rightarrow$ sentence $\rightarrow$ document hierarchy. BCA extends HAN and uses cross-attention. HAN achieves a QWK of **0.748**, while BCA achieves a QWK of **0.800**. Within LSTM based models, there have also been many variations of a two-stage deep-neural network [29], which involves both prompt-aware and prompt-unaware components. Jin *et al.* [29] divide the components according to their linguistic meaning, using separate sub-models of stacked LSTMs for capturing syntax, parts-of-speech, and semantics, respectively. Their best model achieves a QWK of **0.6856**, averaged over all prompts.

All the LSTM-based models mentioned above take at least three days of GPU training time before they can achieve reasonable performance. On the other hand, CNN's cannot be used on their own because even though they select features based on importance, they lead to loss of information and need to be coupled with an LSTM based model that comes with its own set of problems. As detailed in table II, LSTM based models can not achieve more than **0.72** when averaged over all prompts.

In automated essay scoring systems, the state-of-the-art has been achieved by Cozma *et al.* [30] who distinctly approach the problem. The model uses a neural approach to gather word-embeddings called Bag-of-super-word-embeddings (BOSWE). It then uses a kernel-based classifier like the Kernel SVM or Hilbert Space (HSE) in their case to find an appropriate relationship between the embeddings and the output score. The model can achieve a QWK of **0.785** on their best system, which combines BOSWE, HSE, and ν-SVR.

III. The Task

In this experimental setup, given several essays, we have to predict a score for each essay that most agrees with the human rater. The given essay may or may not have a prompt, which is why our model incorporates both a prompt-aware and a prompt-unaware component. Domain experts belonging to two distinct domains have rated the given dataset; the exact nature of their domains is unknown to us. However, this applies to only dataset 2. For this reason and to simplify training and evaluation, we consider the rating of only one domain: domain 1. It becomes part of our training objective and the metric with which we evaluate our system.

[2] https://www.kaggle.com/

A. Dataset

In this paper, all experiments are performed on a dataset borrowed from the official Kaggle competition **The Hewlett Foundation: Automated Essay Scoring** [3], which, despite its age, is the pre-dominant dataset used by automated essay scoring models.

The dataset contains 12,978 essays, segmented into essay sets. There are eight essay sets, and each set has a prompt, type, and average length. Barring essay set **2**, all other essay sets are rated using two raters belonging to the same domain, and the average of the scores are provided to us. It is also the quantity we are predicting. Further description of each set can be found in table I.

TABLE I
ESSAY SET DESCRIPTIONS

Essay Set	Type of Essay	Grade	Essays	Avg. Length
1	persuasive	8	1783	350
2	persuasive	10	1800	350
3	source dependent	10	1726	150
4	source dependent	8	1772	150
5	source dependent	8	1805	150
6	source dependent	8	1800	150
7	persuasive	8	1569	250
8	persuasive	8	723	650

- **Persuasive Essays.** These essays have been written with the intention to *persuade*, to convince the reader to adopt a particular stance on a given topic.
- **Source Dependent Essays**. These essays are analytical essays, and have been written with reference to and in context of given data repository.

B. Metrics

To gauge the performance and robustness of our model, we use the official metric of the competition, the **Quadratic Weighted Kappa**. The quadratic weighted kappa is a measure of agreement between two given vectors. In this particular context, given the scores of $RaterA$ and $RaterB$, we compute the agreement between the two raters as a decimal number between 0 and 1, where 0 means no agreement (random predictions), and 1 means an absolute agreement between the raters. For this experiment, $RaterA$ will be the human rater, and $RaterB$ will be the rating that our model outputs. For essay set 2, we obtained two QWK scores, so we take the average of both.

To calculate QWK, we take the following steps. First, we construct a matrix W according to equation 1.

$$W_{i,j} = \frac{(i-j)^2}{(N-1)^2} \tag{1}$$

where $\mathbf{i}$ and $\mathbf{j}$ are the gold ratings, that have been provided by a human expert, and $\mathbf{N}$ is the range of possible ratings. $\mathbf{O_{i,j}}$ is a matrix computed from $\mathbf{W_{i,j}}$, where each entry is the number of essays that received that particular rating. $\mathbf{E_{i,j}}$ is the outer-product of these two matrices, and finally we compute the κ score as shown in equation 2.

$$\kappa = 1 - \frac{\sum_{i,j} W_{i,j} O_{i,j}}{\sum_{i,j} W_{i,j} E_{i,j}} \tag{2}$$

[3] https://www.kaggle.com/c/asap-aes

IV. PROMPT-UNAWARE COMPONENTS

A. BERT

As previously mentioned in section I, our model implements a BERT encoder in the prompt-unaware phase. The details of the exact implementation are discussed here:

1) Input Embedding Layer: To implement the input embedding layer, we utilize the novel method of WordPiece embeddings introduced by Devlin et al. [16]. Tokenization involves splitting a sentence into sub-words or words, different tokenizers have different rules for tokenization, and the BERT encoder makes use of one of these tokenizers: WordPiece.

WordPiece [31] is very similar to Byte-Pair encoding [32], the compression algorithm turned tokenization algorithm, which acquired first place in the 2016 WMT challenge. However, as opposed to BPE, WordPiece does not rely on frequency alone; it also computes the likelihood of a pair occuring together before merging them. That is, instead of simply relying on the frequency as a parameter, WordPiece estimates the likelihood of a merged word in the training dataset before merging them.

We utilize the embeddings mentioned above in our model. The input layer takes a sentence of length, L_{sent}. If L_{sent} is greater than a pre-specified number, it is truncated. If it is less than L_{sent}, it is padded with the $[PAD]$ token. The sentence is then pre-pended with the $[CLS]$ token. This gives us a vector of tokens for the sentence. The shape of the input vector then becomes (N, L_{sent}). This vector is then converted into a vector of ids by looking up each token in a look-up table

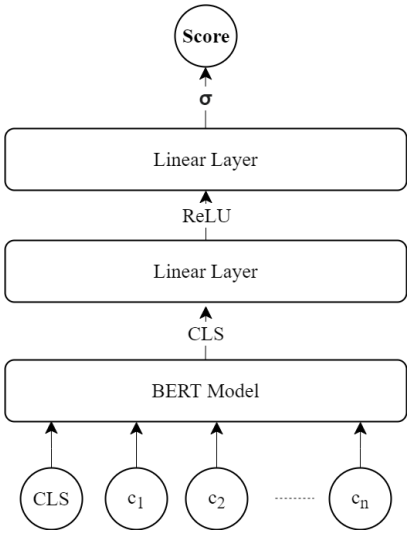

Fig. 1. A flowchart depicting the way in which the BERT model is used to output a score

and replacing it with the corresponding identifying number. In our experiments, the vocabulary size defaults to 30,522.

Since BERT lacks a temporal component, namely LSTMs, we must inject positional information into the model to use that information for language modeling or next sentence prediction.

2) Encoder: BERT is a multi-layer transformer based on the original implementation described in Vaswani *et al.* [12]. A detailed description of transformers is not included here. However, we will briefly discuss the pre-training tasks of BERT and highlight the reasons for BERT's superior performance over vanilla transformer models, despite the similarities in their architectures.

- **Masked Language Modelling:** Language modeling had traditionally been a unidirectional endeavor. Bi-directional language modeling would enable the decoder to inadvertently see the token it was supposed to predict, thus trivializing the task. However, as in conventional linguistics and human speech, it is imperative to know both the left-hand and right-hand context before an accurate prediction can be made. BERT is the first model to achieve bi-directional language modeling by masking out certain words and then asking the model to predict those words. This task has previously been referred to as the *Cloze* task. This creates a few complications, which are mitigated by randomly replacing some tokens with other tokens from the same vocabulary, instead of using the $[MASK]$ token.
- **Next Sentence Prediction:** Essentially, next sentence prediction is a method to predict whether a sentence B is a continuation of a sentence A in a document. This is achieved with the help of a special $token\_type\_ids$ vector where 0 signifies that the sentence is sentence A and one signifies that the sentence is sentence B. The $[CLS]$ token is pre-pended to the sentence, and in the final output vector, the entry corresponding to the $[CLS]$ token is passed through a feed-forward network.

3) Attention Layer: The attention layer is exactly as proposed by Vaswani *et al.* in [12].

- **Scaled Dot Product Attention:** As discussed by Vaswani *et al.* in [12], scaled dot attention has proven to very effective despite its simplicity. We assume that the queries and keys are of dimension d_k and values of dimension d_v. We then compute the dot product attention as shown in figure 2.
- **Multi-Head Attention:** In their experiments Vaswani *et al.* found that accumulating the results of multi-subspace attention networks could provide a lot more context than linear attention. Thus, multi-head attention came into being. This type of attention involves projecting the keys, queries, and values, and then applying dot product attention individually on them. A single layer of attention has a severe drawback. It allows the query to pay attention to the context in only a single subspace. By incorporating multi-attention, we expand attention to multi-subspaces allowing the model to learn across multiple dimensions.

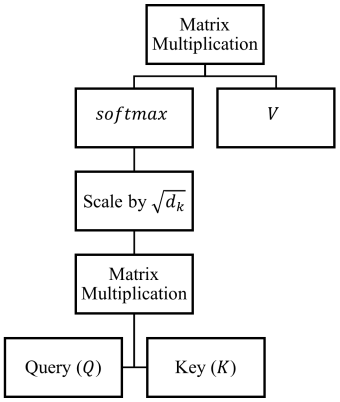

Fig. 2. A visual representation of the scaled dot-product attention used in the attention layer

This allows the model to capture attention in different semantic contexts making it more learned in the long run, despite the increased cost of training.

B. Adapters

Transformer based models have proven their might in most of the natural language processing domain. However, this performance is often accompanied by a huge upshot in number of parameters and training time. To realise the full power of transformers, or models like BERT, even with pre-trained weights, the entire model must be fine-tuned over a certain number of epochs. This entails fine-tuning at least half-a-million parameters in the smallest model (BERT base) and the number can easily reach the billions.

Houlsby *et. al* [17] demonstrate that we can achieve comparable performance on machine comprehension tasks, by learning a couple of thousand parameters instead of a million, by adding **adapter modules** to transformer models. Adapter modules present the best of both worlds: they benefit from the massive knowledge of pre-trained models and achieve high performance with relatively little fine-tuning. Thus, adapter modules achieve both transfer learning and memory efficiency. Figure details the architecture of adapter modules and the place they occupy in the transformer model.

However, this in itself doesn't promise a parameter-efficient model. That is achieved by the bottle-neck architecture of adapter modules. The adapters implement a two-step procedure: d-dimensions $\rightarrow$ m-dimensions $\rightarrow$ non-linearity $\rightarrow$ d-dimensions. This means that the total number of parameters added per layer, including biases, is $2md + d + m$.

As illustrated in figure 3, we add such adapter modules to the BERT model described above, using adapters from AdapterHub [33].

C. Output Layer

To retrieve a single score from the model, we use the *pooled output*, which is a single vector that has captured the entire context. The pooled output has been trained on NER (named entity recognition) and is an space-efficient means of encoding

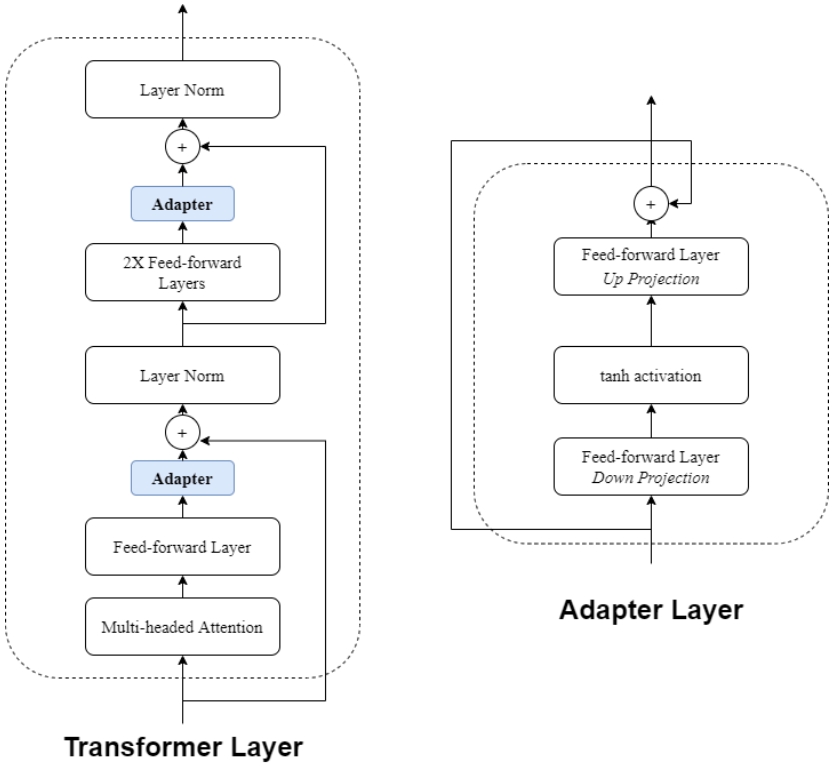

Fig. 3. A visual representation of the adapters layers incorporated in BERT

a large number of tokens. The output is passed through two linear layers and a σ activation to generate a single score.

V. Prompt-Aware Components

A. Bi-directional Attention Flow

As previously mentioned in the section I, we implement a bi-directional attention flow model in our prompt-aware phase. It includes the following layers:

1) Embedding Layer: Given an arbitrarily long sequence of words, we first tokenize the given sequence by using a spaCy tokenizer. Once we have obtained the tokens from the given sequence of words, we obtain their word embedding vectors for both the context (essay) and the query (prompt). This gives us embeddings $c_1, c_2, \ldots, c_n \epsilon R^D$ for the essay, and embeddings $q_1, q_2, \ldots, q_m \epsilon R^D$ for the prompt.

For word embeddings, we use pre-trained 300-dimensional GloVe vectors procured from the NLP group at Stanford. These word vector representations have been obtained after training on a corpus of 840 billion tokens. The pre-training procedure will not be discussed here, but a detailed description can be found in [34].

This gives us word embeddings of dimension H, and these are the word-embeddings we use in the later layers.

2) Encoder Layer: The encoder layer uses a bidirectional LSTM. THe output at any give time-step is the LSTM's hidden cell state, as described in equations 3, 4, and 5.

$$\overrightarrow{l_i} = LSTM(\overrightarrow{l_{i-1}}, x_i)\, \epsilon\, R^H \tag{3}$$

$$\overleftarrow{l_i} = LSTM(\overleftarrow{l_{i-1}}, x_i)\, \epsilon\, R^H \tag{4}$$

$$o_i = [\overrightarrow{l_i}, \overleftarrow{l_i}]\, \epsilon\, R^{2H} \tag{5}$$

where $\overrightarrow{l_i}$ is the (forward) hidden state at time t, $\overleftarrow{l_i}$ is the (backward) hidden state at time t, and o_i is the output at timestep i.

3) Attention Layer: This is where the BiDAF model differs from traditional attention-based LSTM models. Instead of implementing traditional query-to-context attention, it also implements a context-to-query attention layer.

We have with us, hidden states of the context, $c_1, c_2, \ldots, c_n \epsilon R^{2H}$, and hidden states of the query, $q_1, q_2, \ldots, q_m \epsilon R^{2H}$. Firstly, we compute a similarity matrix, $S \epsilon R^{nxm}$, where S_{ij} contains the similarity score for the pair c_i, q_j.

$$S_{ij} = w_{proj}[c_i; q_j; c_i \cdot q_j] \tag{6}$$

Next, we compute the context-to-query attention and query-to-context attention.

- **Context-to-Query Attention**:

$$\bar{S}_{i,:} = softmax(S_{i,:})\, \epsilon R^M \, \forall\, i\, \epsilon\, (1, \ldots, N) \tag{7}$$

$$a_i = \sum_{j=1}^{M} \bar{S}_{i,j} q_j \, \epsilon\, R^{2H} \, \forall\, \epsilon \{1, \ldots, N\} \tag{8}$$

- **Query-to-Context Attention**:

$$\bar{\bar{S}}_{:,j} = softmax(\bar{S}_{:,j}) \forall j \epsilon \{1, \ldots, M\} \quad (9)$$

$$S' = \bar{\bar{S}}\bar{\bar{S}}^T \epsilon R^{NXN} \quad (10)$$

$$b_i = \sum_{j=1}^{N} S'_{i,j} c_j \epsilon R^{2H} \forall i \epsilon \{1, \ldots, N\} \quad (11)$$

4) Modeling Layer: The modeling layer is a novel addition to the BiDAF model. After the attention mechanism has been applied to both query and context, the output is passed into a two-layer deep bi-directional LSTM to encode both query-aware representations of context and context-aware representations of the query. Given input vectors $g_i \epsilon R^{8H}$, the modeling layer computes

$$m_{i,fwd} = LSTM(m_{i-1}, g_i) \epsilon R^H \quad (12)$$

$$m_{i,rev} = LSTM(m_{i+1}, g_i) \epsilon R^H \quad (13)$$

$$m_i = [m_{i,fwd}; m_{i,rev}] \epsilon R^{2H} \quad (14)$$

5) Output Layer: The output layer is supposed to produce a vector of essay scores corresponding to the essays in the given set. Mathematically, the output layer accepts as input the attention layer outputs $g_1, \ldots, g_N \epsilon R^{8H}$ and the modeling layer outputs $m_1, \ldots, m_N \epsilon R^{2H}$, and applies a linear layer individually to both sets, yielding vectors of $(N, L_{sent}, 1)$, which are then added to each other. The output is then passed through a linear layer once again.

VI. Results

Now, we introduce the results of our evaluation and pitch them against the baselines introduced in [25], the deep learning systems analysed in [35]: BERT, XLNet and their ensembles(table II, rows 3-10).

Rows **1** and **2** are baselines, trained on the same dataset mentioned in section III-A. Rows **3** and **4** too are trained on the same dataset mentioned in section III-A, but were trained using 5-fold cross validation i.e. the model was trained on four folds, and evaluated on the remaining fold. This process was repeated for a total of 5 times. Rows **10**, **11**, and **12** were trained in the exact same manner (5-fold validation). The remaining rows **5, 6, 7, 8, 9** were trained just like the model we propose: on the entire dataset with a two held out sets for validation and testing.

The table II shows that our proposed model performs better than the other models, with the average QWK as **0.782**. The proposed model shows is the best performing deep learning model in automated essay scoring. Our model is performing in a better manner for most of the essay prompts. We include the results from Cozma *et al.* [30], strictly for an unbiased comparison, even though it employs more traditional approaches, such as bag-of-words (BOS), and bag-of-super-word-embeddings (BOSWE). Hence, a quantitative comparison, while not possible, has still been attempted.

While it is not entirely possible to pin-point the results, due to the following reasons:

- BERT is a high-performant language model and is trained on huge corpora of data. This is the primary reason, BERT can be used with relatively less fine-tuning. However, the AES dataset contains at most 12,000 essays. It provides essay prevent effective fine-tuning, and causes the robust encoder to forget some of its learnings, something which harms the model. By freezing the parameters of the encoder, and adding another set of parameters (adapters) to learn *how* to use those parameters, we are able to effectively utilise the encoder's knowledge without "resetting it".
- The number of trainable parameters in the adapted BERT model is **98%** less than the un-adapted BERT model. Given the extremely small number of examples in the ASAP dataset, the reduction in the number of trainable parameters is immensely useful to the model, since it has to learn a significantly lower number of parameters.

The modification of BiDAF for this particular use-case, while not state-of-the-art, is an interesting case to analyse. The BiDAF model out-performs all other models on essay set 1. This leads to believe that the grading of the essays in essay set 1, is heavily reliant on the prompt. That is, when the essays are graded independent of the prompts, the grading does not "agree" much with the gold grades. However, when we use a prompt-aware component to grade the essays, we are able to achieve an improvement of 5% over the best-known model. A deep dive into the dataset supports our assumption i.e. this dataset contains adversarial essays (essays that have been written with the express purpose of defeating the system). A few essays in set 1, while largely coherent linguistically, are not aligned with the prompt of that set, as shown below.

> ...this, their safety is at risk. Finally, people should stop using computers as much as they are now, in order to keep their safety safe. **Furthermore, enjoying nature is a must, in our world full of immense landscapes, wide waters, dense forests, and lush valleys.** @CAPS1 you depend on your computer to teach you about the world, you should cease ...

The highlighted line is not aligned to the prompt of essay set 1, and should result in a penalty; a fact that a prompt-unaware system is unable to deduce. Another noteworthy point of the BiDAF model, is its superior performance on essay sets 6 and 8, coming in second to the adapted BERT models. An analysis of the essays from these sets too reveal a strong inter-dependency between the essay and the prompt.

VII. Conclusion

In this paper, we propose a transformer-based model, retro-fitted with adapters, to solve the task of automated essay scoring. We pitch this model against all models we gathered

TABLE II
RESULTS: COMPARISON OF MODELS

ID	Systems	Prompts								
		1	2	3	4	5	6	7	8	QWK
1	EASE (SVR)	0.781	0.621	0.630	0.749	0.782	0.771	0.727	0.534	0.699
2	EASE (BLRR)	0.761	0.606	0.621	0.742	0.784	0.775	0.730	0.617	0.705
3	LSTM [25]	0.775	0.687	0.683	0.795	0.818	0.813	0.805	0.594	0.746
4	LSTM + CNN [25]	0.821	0.688	0.694	0.805	0.807	0.819	0.808	0.644	0.760
5	BERT [35]	0.792	0.679	0.715	0.800	0.805	0.805	0.785	0.595	0.748
6	XLNet [35]	0.776	0.680	0.692	0.806	0.783	0.793	0.786	0.628	0.743
7	BERT Ensemble [35]	0.802	0.672	0.708	0.815	0.806	0.814	0.804	0.597	0.752
8	XLNet Ensemble [35]	0.804	0.685	0.7009	0.795	0.799	0.805	0.800	0.597	0.748
9	BERT + XLNet Ensemble [35]	0.807	0.696	0.703	0.819	0.808	0.815	0.806	0.604	0.757
10	HISK and ν-SVR [30]	0.836	0.724	0.677	0.821	0.830	0.828	0.801	0.726	0.780
11	BOSWE and ν-SVR [30]	0.788	0.689	0.667	0.809	0.824	0.824	0.766	0.679	0.756
12	HISK+BOSWE and ν-SVR [30]	0.845	**0.729**	0.684	0.829	0.833	0.830	0.804	0.729	0.784
13	BiDAF [18]	**0.844**	0.686	0.665	0.723	0.760	0.817	0.746	0.724	0.746
14	Proposed model	0.743	0.674	**0.718**	**0.884**	**0.834**	**0.842**	**0.819**	**0.744**	**0.785**

from existing literature, and show that our model outperforms all given models on **five** of **eight** essay prompts, and comes in a close second on the remaining **three** prompts. We incorporate a BERT-based backbone into our model, and freeze **99%** of parameters in the backbone; effectively training only a million parameters in total. This reduces training costs by **99%** and still manages to out-perform the BERT model on the ASAP dataset on all prompts. Thus, we are able to introduce a lightweight model with **99%** less trainable parameters than conventional language models, and still outperform them all.

As an extensive study, we also re-purpose models that are traditionally used for other tasks, such as the BiDAF [18] model for question-answering, and use them for the task of automated essay scoring by modifying their architecture as detailed in section V-A. Even though this model falls short of other algorithms, it proves it worth in detecting adversarial essays as discussed in section II. We even peruse existing literature and present all algorithmic attempts to solve the task of Automated Essay Scoring, along with their training methodologies and results for better quantitative analysis of our presented model's performance on the Automated Student Assessment Prize data set.

REFERENCES

[1] M. Chodorow and J. Burstein, "Beyond essay length: Evaluating e-rater®'s performance on toefl® essays," *ETS Research Report Series*, vol. 2004, no. 1, pp. i–38, 2004.

[2] B. Beigman Klebanov, C. Stab, J. Burstein, Y. Song, B. Gyawali, and I. Gurevych, "Argumentation: Content, structure, and relationship with essay quality," in *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 70–75.

[3] S. Katyal, S. Kumar, R. Sakhuja, and S. Gupta, "Object detection in foggy conditions by fusion of saliency map and yolo," in *2018 12th International Conference on Sensing Technology (ICST)*. IEEE, 2018, pp. 154–159.

[4] S. Kumar and M. Kumar, "Predicting customer churn using artificial neural network," in *Engineering Applications of Neural Networks*, J. Macintyre, L. Iliadis, I. Maglogiannis, and C. Jayne, Eds. Cham: Springer International Publishing, 2019, pp. 299–306.

[5] A. Bhowmik, S. Kumar, and N. Bhat, "Eye disease prediction from optical coherence tomography images with transfer learning," in *Engineering Applications of Neural Networks*, J. Macintyre, L. Iliadis, I. Maglogiannis, and C. Jayne, Eds. Cham: Springer International Publishing, 2019, pp. 104–114.

[6] S. Kumar and M. Kumar, "A study on the image detection using convolution neural networks and tenserflow," in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2018, pp. 1080–1083.

[7] K. Singh and A. S. Parihar, "Variational optimization based single image dehazing," *Journal of Visual Communication and Image Representation*, vol. 79, p. 103241, 2021.

[8] A. S. Parihar, K. Singh, H. Rohilla, and G. Asnani, "Fusion-based simultaneous estimation of reflectance and illumination for low-light image enhancement," *IET Image Processing*, vol. 15, no. 7, pp. 1410–1423, 2021.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997.

[10] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[11] M. Bianchini and F. Scarselli, "On the complexity of neural network classifiers: A comparison between shallow and deep architectures," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 8, pp. 1553–1565, 2014.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[13] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," *CoRR*, vol. abs/1804.07461, 2018. [Online]. Available: http://arxiv.org/abs/1804.07461

[14] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Superglue: A stickier benchmark for general-purpose language understanding systems," *CoRR*, vol. abs/1905.00537, 2019. [Online]. Available: http://arxiv.org/abs/1905.00537

[15] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. of NAACL*, 2018.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[17] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 2790–2799.

[18] M. J. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *CoRR*, vol. abs/1611.01603, 2016.

[19] V. Vapnik, S. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation and signal processing," in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. Jordan, and T. Petsche, Eds., vol. 9. MIT Press, 1997.

[20] Y. Attali and J. Burstein, "Attali, y., & burstein, j. (2006). automated essay scoring with e-rater® v.2. journal of technology, learning, and assessment, 4(3)." *Journal of Technology, Learning, and Assessment*, vol. 4, 02 2006.
[21] "Study of variants of extreme learning machine (ELM) brands and its performance measure on classification algorithm," *June 2021*, vol. 3, no. 2, pp. 83–95, Jun. 2021. [Online]. Available: https://doi.org/10.36548/jscp.2021.2.003
[22] M. Tripathi, "Sentiment analysis of nepali COVID19 tweets using NB, SVM AND LSTM," *September 2021*, vol. 3, no. 3, pp. 151–168, Jul. 2021. [Online]. Available: https://doi.org/10.36548/jaicn.2021.3.001
[23] S. Yasir Babiker Hamdan, "Construction of statistical svm based recognition model for handwritten character recognition," *December 2021*, vol. 3, no. 2, pp. 92–107, 2021.
[24] V. T, "COMPARATIVE STUDY OF CAPSULE NEURAL NETWORK IN VARIOUS APPLICATIONS," *Journal of Artificial Intelligence and Capsule Networks*, vol. 01, no. 01, pp. 19–27, Sep. 2019. [Online]. Available: https://doi.org/10.36548/jaicn.2019.1.003
[25] K. Taghipour and H. T. Ng, "A neural approach to automated essay scoring," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1882–1891.
[26] F. Nadeem, H. Nguyen, Y. Liu, and M. Ostendorf, "Automated essay scoring with discourse-aware neural models," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 484–493.
[27] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 1480–1489.
[28] J.-B. Remy, A. J.-P. Tixier, and M. Vazirgiannis, "Bidirectional context-aware hierarchical attention network for document understanding," 2019.
[29] C. Jin, B. He, K. Hui, and L. Sun, "TDNN: A two-stage deep neural network for prompt-independent automated essay scoring," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 1088–1097.
[30] M. Cozma, A. Butnaru, and R. T. Ionescu, "Automated essay scoring with string kernels and word embeddings," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 503–509.
[31] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016, cite arxiv:1609.08144.
[32] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *CoRR*, vol. abs/1508.07909, 2015.
[33] J. Pfeiffer, A. Rücklé, C. Poth, A. Kamath, I. Vulić, S. Ruder, K. Cho, and I. Gurevych, "Adapterhub: A framework for adapting transformers," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020): Systems Demonstrations*. Online: Association for Computational Linguistics, 2020, pp. 46–54.
[34] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://www.aclweb.org/anthology/D14-1162
[35] P. U. Rodriguez, A. Jafari, and C. M. Ormerod, "Language models and automated essay scoring," *CoRR*, vol. abs/1909.09482, 2019.