Beyond Canonical Fine-tuning: Leveraging Hybrid Multi-Layer Pooled Representations of BERT for Automated Essay Scoring

Eujene Nikka V. Boquio, Prospero C. Naval Jr.
University of the Philippines Diliman
Quezon City, Philippines
{evboquio, pcnaval}@up.edu.ph

Abstract

The challenging yet relevant task of automated essay scoring (AES) continuously gains attention from multiple disciplines over the years. With the advent of pre-trained large language models such as BERT, fine-tuning those models has become the dominant technique in various natural language processing (NLP) tasks. Several studies fine-tune BERT for the AES task but only utilize the final pooled output from its last layer. With BERT's multi-layer architecture that encodes hierarchical linguistic information, we believe we can improve overall essay scoring performance by leveraging information from its intermediate layers. In this study, we diverge from the canonical fine-tuning paradigm by exploring different combinations of model outputs and single- and multi-layer pooling strategies, as well as architecture modifications to the task-specific component of the model. Using a hybrid pooling strategy, experimental results show that our best essay representation combined with a simple architectural modification outperforms the average QWK score of the basic fine-tuned BERT with default output on the ASAP AES dataset, suggesting its effectiveness for the AES task and potentially other long-text tasks.

Keywords: AES, BERT, fine-tuning, representation learning

1. Introduction

In an educational setting, essay writing is an important tool to help in the development of students' language proficiency skills, as well as higher-order thinking skills such as critical thinking and idea synthesis. Over the years, several efforts have been made to automate the essay scoring process to reduce costs and provide instantaneous feedback. This task of Automated Essay Scoring (AES) aims to assign a score to an essay written on a certain topic or prompt based on its overall quality or different writing criteria. Earlier works leverage handcrafted features to represent the different measurable properties of essays such as essay length and grammatical errors that contribute to their quality. However, manual feature engineering is an expensive and elaborate process and features extracted are usually limited to lower-level textual features since complex and higher-level features that capture the semantics, discourse, and pragmatics (Dong et al., 2017) are difficult to extract (Beseiso and Alzahrani, 2020).

With the rise of neural networks, end-to-end models for learning good essay representations that can capture deep semantic features have been developed. Existing works are usually based on Convolutional Neural Networks (CNNs) and Long Short Term Memory (LSTM) networks (Taghipour and Ng, 2016; Dong and Zhang, 2016; Dong et al., 2017). While these models have obtained promising performance for the AES task without laborious feature engineering, they require large quantities of labeled data for training to obtain good performance. This poses a challenge in the task of AES since it is difficult and expensive to obtain a large amount of human-annotated essays for each specific essay prompt. To address this challenge, researchers utilize transfer learning, which allows a pre-trained model to be adapted to fit different downstream tasks without the need to build and train a new model from scratch.

In recent years, there has been a surge of research interest in transfer learning due to its improved performance and representation ability in a variety of downstream tasks. With the advent of pre-trained language models (PLMs), knowledge learned from a large corpus of text data can be transferred to many downstream tasks by fine-tuning, which is done by simply replacing the output layer of the model with a task-specific layer to optimize task-specific objectives for only a few training epochs. This pre-training-then-fine-tuning paradigm has since become the common practice in the field of natural language processing (NLP) and forms the basis of state-of-the-art results on many tasks (Devlin et al., 2018; Yang et al., 2019).

One of the most effective PLMs is Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018). Fine-tuning pre-trained BERT has led to significant boosts in the state-of-the-art performance for a variety of NLP tasks (Wang et al., 2018), showing its remarkable ability to learn universal language representations. BERT's striking success can be explained by its use of a novel language modeling approach and a multi-layer bidirectional Transformer (Vaswani et al., 2017), which helps to capture the context in long sequences of words. This makes it very attractive when it comes to AES, as it is one of the more challenging NLP tasks due to relatively longer texts as compared to other tasks like machine translation. Furthermore, with the help of BERT's use of the self-attention mechanism, it is able to capture longer time dependencies introduced by the length and structure of essays and focus on the most relevant features for the task at hand.

Existing AES approaches using pre-trained BERT utilize the output of BERT's final layer as the essay representation. This output corresponds to the activation of the special [CLS] token that summarizes information for each essay using the self-attention mechanism. However, leveraging output from a single layer could restrict the power of obtained representation since the semantic knowledge learned at the intermediate layers is ignored (Yang and Zhao, 2019; Song et al., 2020). BERT's multi-layer architecture allows it to encode a hierarchy of linguistic information for the specific task: it learns more general and transferable (surface and syntactic) features in the lower layers and more task-specific (semantic) information in the top layers (Hao et al., 2020; Jawahar et al., 2019; Peters et al., 2019). We believe that we can utilize the information learned from intermediate layers to learn better essay representations that capture more semantic information.

In this study, we investigate different ways of fine-tuning BERT by utilizing its intermediate layers to improve its performance on the task of AES. We first perform initial experiments to find good base hyperparameters to be used for the rest of the experiments. For our main experiments, we focus on finding the best essay representation for the essay scoring task. We do this by utilizing BERT's intermediate layers and testing the usage of combinations for different BERT model outputs and pooling strategies. We also test these representations in combination with different model architectures for the task-specific component of the model.

Our main contributions are as follows:

1. To the best of our knowledge, this is the first study to go beyond the traditional way of fine-tuning specifically for the AES task by finding other essay representations and modifying the task-specific architectures.

2. We examine the potential of utilizing BERT intermediate layers in combination with different pooling strategies (for single and multiple layers) for the AES task. Our best essay representation uses a hybrid method that combines different single- and multi-layer pooling strategies.

3. Experimental results show that compared to using the default BERT last-layer output, we can greatly improve performance by pooling information from all 12 layers of the BERT-base model, or even just a subset of the layers. Moreover, we present simple modifications to the task-specific component of the model that can be beneficial for the essay scoring task.

4. We also perform some hyperparameter tuning to establish good initial hyperparameters. We also consider the overlooked SGD optimizer over the default Adam optimizer for BERT and show that it can achieve better overall test performance.

2. Related Literature

2.1. Automated Essay Scoring

Traditional methods construct handcrafted features to be fed as input to statistical methods. Recently, researchers have leveraged advances in deep learning for AES. Deep neural networks such as LSTM and CNN were shown to achieve competitive performance compared to traditional approaches. Taghipour and Ng (2016) was the first to do this for the AES task by utilizing a single-layer LSTM. Since then, several architectures and techniques were proposed to improve performance (Dong and Zhang, 2016; Dong et al., 2017). These works use pre-trained word embeddings such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), where a word's representation is the same in all contexts. For a document-level task such as AES, it is important to capture context and long-distance relationships between words.

Contextual word representations obtained from large neural language models such as BERT (Devlin et al., 2018) have become dominant as they encode useful representations that are adjusted based on their context. Following BERT's success in many NLP tasks, it has now been used for AES (Rodriguez et al., 2019; Mayfield and Black, 2020; Yang et al., 2020; Wang et al., 2022; Sun et al., 2022). Rodriguez et al. (2019) compares BERT and XLNet (Yang et al., 2019) and other traditional methods for the AES task. Yang et al. (2020) propose a multi-loss model that fine-tunes BERT for the essay scoring and ranking task. Wang et al. (2022) uses BERT to jointly learn a multi-scale essay representation, improving over traditional deep methods.

2.2. Utilizing BERT Intermediate Layers

AES methods that fine-tune BERT use the output of the final classification ([CLS]) token as essay representation (Rodriguez et al., 2019; Yang et al., 2020; Sun et al., 2022). However, due to BERT's multi-layer architecture, the information captured specializes for the language modeling tasks as we approach its last layers (Hao et al., 2020; Jawahar et al., 2019; Peters et al., 2018; Liu et al., 2019). Thus, the single final layer might not always be the best representation. Some studies utilize intermediate layers of BERT for other tasks and showed performance improvements. In BERT's original paper (Devlin et al., 2018), combinations of features from different layers are used for a named entity recognition task and found that concatenation of the last four layers achieved the best performance. In (Song et al., 2020), multi-layer representations of the [CLS] token are integrated with LSTM and attention pooling, which both showed improvements in sentiment analysis and natural language inference (NLI) tasks. Tenney et al. (2019) achieved significant performance gains that are attributed to the intermediate layers containing the most relevant features. These studies motivated us to exploit the representations learned by intermediate layers for AES.

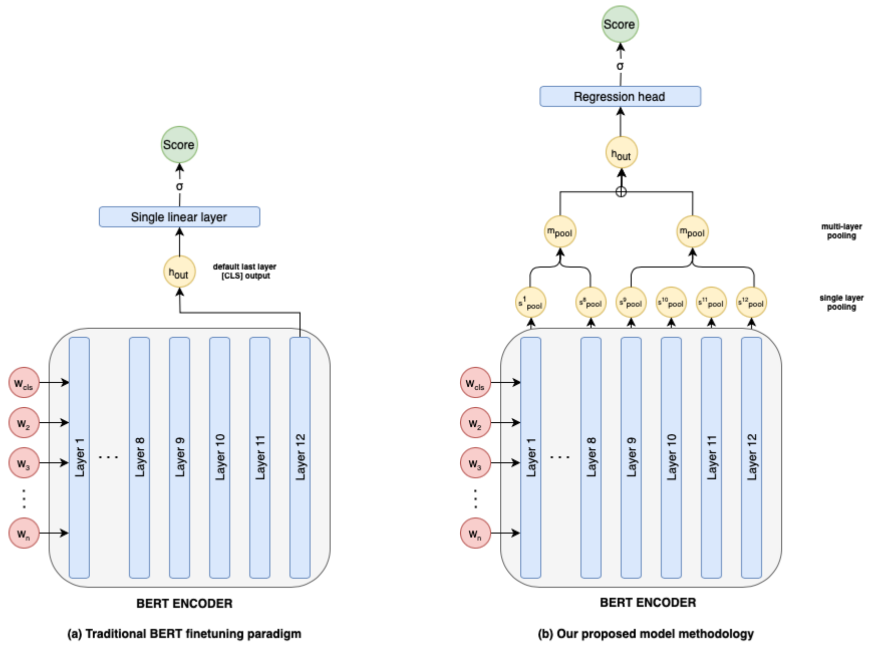

Figure 1: Comparison of traditional fine-tuning paradigm for BERT and our proposed model for utilizing BERT intermediate layers. Different model outputs (obtained from single or multi-layer pooling or their combination), denoted by yellow circles, are used for our representation learning experiments.

3. Methodology

Figure 1 shows our proposed general model architecture. We differ from the traditional fine-tuned BERT in 2 main ways: the chosen model output as essay representation and the task-specific or regression head of the model. We experiment with model outputs from different BERT layers using several pooling strategies, which are applied to either a single layer or a set of multiple layers. Our chosen model applies different single-layer and multi-layer pooling strategies on all layers of the BERT encoder. For the model's regression head, we test 2 architecture modifications alongside the common way of appending a single linear layer to BERT.

3.1. Background: BERT

BERT (Devlin et al., 2018) employs a Transformer architecture (Vaswani et al., 2017) with a multi-head self-attention mechanism in each layer which can help capture long-distance relationships between words. It is pre-trained on BookCorpus (Zhu et al., 2015) and English Wikipedia (in total 3.3B words) with the masked language modeling (MLM) and next sentence prediction (NSP) objectives. By leveraging bi-directional self-attention for the pre-training tasks, it can produce contextual representations that fuse information from the left and right context jointly in all layers.

To use BERT on downstream tasks, preprocessing of the input text is required. Specifically, a special token ("[CLS]") is prepended to each text. Typically, the model output corresponding to this token is used as the aggregated text representation that is passed to the task-specific layer for fine-tuning to fit the target task.

3.2. Dataset

We use the Automated Student Assessment Prize (ASAP) dataset[1], which was the official dataset used in the ASAP competition in 2012. This benchmark dataset for AES consists of about 13,000 English essays written by students in Grades 7 to 10 across eight different prompts. A train/validation/test split of 60/20/20 is used. Details of the dataset are shown in Table 1. The token length refers to the average length of the WordPiece tokens for the essays in each prompt.

Set	Essay Type	# Essay	Ave Len	Score Range	Token Len
1	Argumentative	1785	350	2 - 12	649
2	Argumentative	1800	350	1 - 6	704
3	Source-dependent	1726	150	0 - 3	219
4	Source-dependent	1772	150	0 - 3	203
5	Source-dependent	1805	150	0 - 4	258
6	Source-dependent	1800	150	0 - 4	289
7	Narrative	1569	300	0 - 30	371
8	Narrative	723	650	0 - 60	1077

Table 1: Description of the ASAP dataset.

3.3. Experimental Setup

Preprocessing of the essays is done by converting all characters to lowercase, removing special characters, and performing tokenization using WordPiece tokenizer (Wu et al., 2016), which adds two special tokens: [CLS] to the beginning of each input and [SEP] to separate. Since BERT can only take a maximum sequence length of 512 tokens, we truncate essays longer than

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

510 words (taking the 2 special tokens into account). We also perform padding to shorter essays.

For the implementation, we use the pre-trained weights of the ”bert-base-uncased” model implementation of the Huggingface transformers library (Wolf et al., 2019), which consists of 12 layers that each has a hidden size of 768 and 12 attention heads. We perform the traditional way of fine-tuning language models by adding a single linear output layer, with a sigmoid activation function to the BERT-base model, which we refer to as BERT-finetune. We use the last hidden state of the CLS token as the default representation of each essay. The sigmoid activation function projects the output vector x of the BERT model to a scalar value in the range [0,1] as shown in Equation 1, where W is a weight matrix and b is a bias value.

$$s(x) = sigmoid(W \cdot x + b) \tag{1}$$

The reference scores are normalized so they are within the range [0,1]. The predicted scores by the AES system are scaled back to the original range of the scores during testing. The scoring task is treated as a regression problem and thus, the Mean Square Error (MSE) (Equation eq:mse) between the corresponding original ($\hat{y}$) and predicted (y) score for N essays is used as the loss function. All models are trained for 100 epochs, but the model with the best validation QWK is chosen. All experiments are implemented using PyTorch.

$$MSE(\hat{y_i}, y_i) = 1N \sum_{i=1}^{N} (\hat{y_i} - y_i)^2 \tag{2}$$

To evaluate the performance of the method, we use the quadratic weighted kappa (QWK) metric to measure the agreement between two raters. This was the official evaluation metric used in the ASAP competition and has since been the standard metric for AES.

3.4. Initial Experiments

Using the default lhs-cls outputs, initial experiments were done on Prompt 1. These include hyperparameter tuning to find a good starting point for the hyperparameters to be used for the next experiments. The hyperparameters tested are shown in Table 2.

Hyperparameters	**Values tested**
Optimizer	**SGD**, Adam, **AdamW**
Scheduler	None, **Linear decay**, Cosine, Polynomial decay
Learning rate	SGD: 0.001, **0.01**, **0.03**, 0.05, 0.08, 0.1, Adam/AdamW: $3e^-6$, e^-5, $5e^-5$, $1e^-4$
Dropout rate	0, **0.1, 0.3**, 0.5

Table 2: Hyperparameters used for initial experiments. The chosen hyperparameters are in bold.

As we can see from the table, we explored different hyperparameters, one of which is the choice of optimizer: Adam, AdamW (Adam with weight decay) (Loshchilov and Hutter, 2017), and SGD (stochastic gradient descent) with momentum. Adam optimizers are typically used with BERT in previous papers, especially those that fine-tune BERT for AES. One reason for this is the original BERT was also trained using Adam optimizer, which has become the default optimizer for most applications due to its fast convergence time and need for fewer parameters during training. Adam mainly differs from SGD by dynamically adjusting individual learning rates for its parameters instead of maintaining a single learning rate, resulting in faster convergence. It generally gives good and acceptable results in most applications, and thus, not much effort is given to tuning or optimizing its performance, or even exploring other optimizers. However, SGD is recently being compared to Adam optimizer in various applications due to its better generalization performance on the test set despite taking longer to converge (Keskar and Socher, 2017; Zhou et al., 2020). Thus, we explore these optimizer choices through hyperparameter tuning. For SGD, we use a momentum of 0.9.

3.5. Representation Learning

BERT’s default output representation is as follows: given a sequence of n tokens $\{w_1, \ldots, w_n\}$, which include special tokens and the words in an input essay, BERT encodes the sequence into the contextualized representation $R \in R^{n \times d}$ given by:

$$R = BERT(\{w_1, \ldots, w_n\}) \tag{3}$$

where R is the output of the last layer of the BERT encoder and d is the hidden size. R corresponds to the first token ([CLS]) of the last hidden state. However, other types of model outputs from intermediate layers could also provide a better representation of the essays for the AES task.

In this study, we explore two main outputs of the BERT model. The first is the last hidden state (lhs), which is the sequence of hidden states at the last layer of the model, and the second is hidden states (hs), which contain the aggregation of hidden states of multiple layers and sequences. We consider different variations for the lhs and hs outputs. For the lhs output, we not only consider the last layer (lhs), but also the second-, third-, and fourth-to-the-last layers, denoted by 2lhs, 3lhs, and 4lhs, respectively. For the hs output, we consider the aggregation of the hidden states of the last 4 layers (gl4 - get last 4), the first 8 layers (gf8 - get first 8), and all 12 layers (ahs - all hidden states).

Moreover, various pooling strategies for the lhs and hs outputs are considered, as shown below. We refer to the pooling of lhs outputs as single-layer pooling, while pooling the hs outputs of a set of multiple layers is referred to as multi-layer pooling. Different combinations of model representations/output and pooling strategies are used. For example, the default representation of getting the CLS embedding of the last hidden state is denoted as lhs-cls. The second-to-last hidden state with attention pooling is denoted as 2lhs-att while concatenating the last 4 hidden states is denoted as gl4-concat, and so on.

3.5.1. Single-layer Pooling Strategies

Pooling strategies for lhs outputs involve applying the pooling operation to all tokens in a layer K.

CLS embedding (cls) is obtained by taking h^K_{cls} of a layer K. For instance, the CLS embedding of the third-to-the last layer (3lhs-cls) is h^{10}_{CLS}.

Mean pooling (mean) averages the hidden states of all n token embeddings in a layer. We ignore the [PAD] token by utilizing the attention masks.

$$\mathrm{h}^K_{mean} = 1n \sum_{i=1}^{n} H^K_i$$

Max pooling (max) takes the maximum across n token embeddings. Attention masks are also used.

$$\mathrm{h}^K_{max} = \max_{i=1,\dots,n} H^K_i$$

Mean-max pooling (mm) finds both mean and max pooling embeddings and concatenates them.

$$\mathrm{h}^K_{mm} = h^K_{mean} \parallel h^K_{max}$$

Attention pooling (att) uses a dot-product attention operation on all token embeddings for a layer K to learn their contribution. This is given by:

$$\mathrm{a}_i = \tanh\left(W^K_a \cdot h^K_i + b_a\right)$$
$$\alpha_i = e^{w_\alpha \cdot a_j} \sum e^{w_\alpha \cdot a_j}$$
$$\mathrm{h}^K_{att} = \sum \alpha_i{}^K_i$$

where W_a is the weight matrix, w_α is the weight vector, b_a is the bias vector, and a_i and α_i are the attention vector and attention weight for the i^{th} token respectively.

Conv1d pooling (conv) h^K_{conv} uses 1-dimensional convolution layers (Kiranyaz et al., 2021) that slide across all n tokens to extract relevant features and ignore unwanted ones. We use a kernel size of 2 tokens and a padding size of 1.

3.5.2. Multi-layer Pooling Strategies

We denote the hidden states of the CLS token of BERT with L layers as $h_{CLS} = \{h^1_{CLS}, h^2_{CLS}, \dots, h^L_{CLS}\}$. By default, pooling strategies for hs outputs (hs_pool) are applied on the CLS embeddings from a set of layers S (e.g. gl4-concat uses concatenates the CLS embeddings of the last 4 layers, or $S = \{9, 10, 11, 12\}$). When using single-layer pooling other than CLS embedding, we add the pooling method after the notation (e.g. gl4-att-concat concatenates output obtained from attention pooling from each of the last 4 layers).

Mean pooling (mean-hs) involves taking the mean pooling of the outputs for each layer in a combination of layers and stacking them together.

$$\mathrm{h}_{mean-hs} = 1|S| \sum_{l \in S} h^l_{cls}$$

Concatenate pooling (concat) simply concatenates outputs from multiple layers into one. For example, gl4-concat concatenates the outputs from the last 4 layers.

$$\mathrm{h}_{concat} =_{l \in S} h^l_{cls}$$

Weighted layer pooling (wl) takes the weighted mean of the token embeddings of layers in S.

$$\mathrm{h}_{wl} = \sum w_\alpha \cdot h^l_{cls} \sum w_\alpha$$

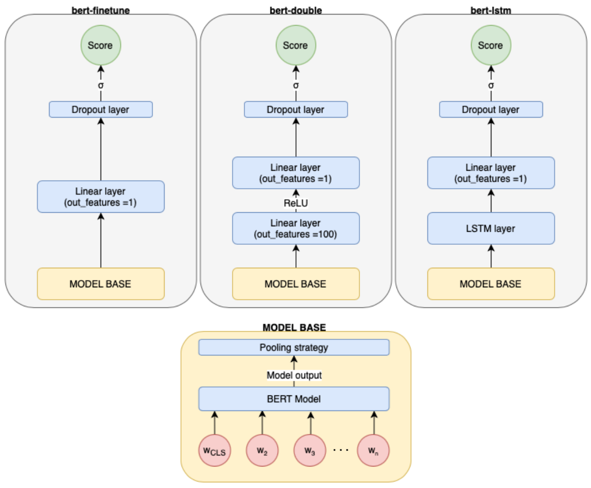

Figure 2: Model Architectures tested

3.6. Model Architectures

Three different model architectures were tested, as shown in Figure 2, all of which use the BERT-base model as the backbone. Only the regressor (task-specific layers) are changed. The default way of fine-tuning (adding an output layer) is used in bert-finetune. In bert-double, 2 linear layers are added with a ReLU (Rectified Linear Unit activation function (LeCun et al., 2015) between them. bert-lstm uses a single LSTM layer to pool the BERT output before it is passed to the final layer. By default, all models with different output and pooling combinations are trained with these hyperparameters: batch size of 32, dropout rates of 0, 0.1, and 0.3, linear scheduler with 0 warmup steps, and gradient clipping with max norm of 1.0. We initialize model parameters to the pre-trained values for use on the task-specific dataset.

3.7. Baseline Models

The baseline models for comparison are described below:

EASE (Phandi et al., 2015) employs regression techniques using handcrafted features, such as Support Vector Regression (SVR) and Bayesian Linear Ridge Regression (BLRR) . It is the best open-source system[2] that joined the ASAP competition, ranking third place overall.

LSTM-based deep neural networks are studied in (Taghipour and Ng, 2016). In particular, we compare with the vanilla LSTM network and the LSTM + CNN network that combines CNN and LSTM ensembles over 10 runs in their experiments.

Simpler transformer-based models are studied in (Rodriguez et al., 2019), with results for using BERT and XLNet (Yang et al., 2019) models, as well as ensembles for BERT, XLNet, and their combination.

[2]http://github.com/edx/ease

Hybrid models that use both low-level features and high-level semantic feature representation methods are shown in (Cozma et al., 2018). Results for HISK and ν-SVR, BOSWE and ν-SVR, and HISK+BOSWE and ν-SVR are reported.

Parameter-Efficient Transformer (Sethi and Singh, 2022) use transformer-based pre-trained language model with adapter models to reduce number of trainable parameters.

Self-supervised methods from (Cao et al., 2020) use two self-supervised tasks and a domain adversarial training technique, becoming the first work that employs pre-trained language model to outperform LSTM-based methods. Their study report results on hierarchical LSTM model and BERT as their base encoders, which are HA-LSTM+SST+DAT and BERT+SST+DAT respectively.

R^{2}BERT (Yang et al., 2020) employs a multi-loss function that combines regression and ranking to fine-tune BERT model.

4. Results and Discussion

4.1. Initial Experiments

Initial hyperparameter tuning was performed to determine a good combination of the optimizer, learning rate, and dropout rates. There is not much of a significant difference between the performance of the Adam and AdamW optimizer, but the latter is slightly better. Five configurations are then chosen and trained on all the prompts using the three model architectures. The results are shown in Table 3. Using the default bert-finetune model, we can see that using SGD with a 0.01 learning rate achieves the best average QWK. For bert-double and bert-lstm, SGD lr=0.03 achieves the best average QWK.

On average, we notice SGD optimizer to have better scoring performance on all essay prompts using all 3 model architectures. Although its loss curves confirm slower convergence compared to the Adam optimizers, we use SGD for the next experiments du.l,e to its ability to better generalize on the test set. Our results add to the studies that confirm that SGD with momentum can be more beneficial than Adam or AdamW when it comes to generalization performance, at least for AES using the ASAP dataset.

MODEL	LR	lhs-cls	lhs-att	lhs-mean	lhs-max	lhs-mm	lhs-conv
bert-finetune	0.01	**0.8351**	0.8336	0.8102	0.8324	**0.8364**	**0.8059**
	0.03	0.8152	**0.8398**	**0.8205**	**0.8333**	0.8240	0.8015
bert-double	0.01	**0.8259**	**0.8317**	**0.8327**	0.8301	**0.8388**	0.8314
	0.03	0.8202	0.8231	0.8238	**0.8351**	0.8206	**0.8381**
bert-lstm	0.01	**0.8259**	0.8095	0.8203	0.8299	0.8368	0.8155
	0.03	0.8202	**0.8183**	**0.8349**	**0.8349**	**0.8446**	**0.8321**

Table 4: QWK scores using different pooling strategies on the last layer. All models have SGD as the optimizer. The better values for each model and pooling strategies are in bold.

4.2. Main Results

4.2.1. Representation Learning

We perform experiments by training the model architectures on Prompt 1 using the different hyperparameter configurations and combinations of model output (lhs and hs outputs) and pooling strategies to get an idea of which pooling methods work well for each model. In Table 4, we report the results of using different pooling strategies on the final layer only. For the default bert-finetune model, conv and mean pooling do not give the best scoring performance while the rest are good pooling methods for the last layer. With att pooling, we achieve the best test QWK for this model. For bert-double, conv pooling works well, while cls embedding does not work as well as in the previous model. The best scoring performance is achieved with mm pooling. Finally, for the bert-lstm model, mm pooling also achieved the best QWK score, with mean pooling also getting good results. Also, att pooling and cls embedding do not work as well as the others. We also observe that a higher learning rate (0.03) works better for most of the representations. Overall, mm pooling does a great job of capturing the essay representations at the last layer across the models. The best QWK score is achieved with the bert-lstm model, mm pooling, and learning rate of 0.03. Thus, we use a 0.03 learning rate.

Next, we present the test QWK scores using the output from the last 4 layers, for each model and pooling strategy in Table 5. Since we want to utilize multiple BERT layers, we compare the average QWK scores for all 4 layers. For the bert-finetune model, the pooling method that got the best average performance is att pooling. We also notice that att and mean pooling are the only methods to surpass the average performance using the default cls output. It can be seen that for 3lhs and 4lhs, the cls output got a much better score than the other pooling methods. This shows that for the default fine-tuning technique, the CLS output from only the last layer may not be the best essay representation and it is important to consider other layers too.

For the bert-double model, all other pooling methods improved over the cls embedding. Moreover, the pooling method with the best performance is conv pooling, but we can also see that max and mm pooling achieved good scores. Lastly, for the bert-lstm model, the pooling method with the best average performance is mm pooling, with max pooling as a good pooling method too. Just as with the bert-double model, all the other pooling methods got improved average performance compared to the cls embedding, which shows that considering different pooled outputs can obtain better essay representations. Out of all the configurations, the best average performance for all 4 layers is achieved with the bert-lstm model using mm pooling. We attribute this to the ability of the LSTM layer to learn and remember long sequences of inputs since the output of mm pooling is twice as long as the other pooled outputs. Based on the table, we note the improvement of

MODEL	CONFIG	SET 1	SET 2	SET 3	SET 4	SET 5	SET 6	SET 7	SET 8	AVE
bert-finetune	AdamW lr=3e^{-6}	**0.8480**	0.6361	0.6774	0.8085	0.8024	0.8179	0.8022	0.7678	0.7700
	AdamW lr=5e^{-5}	0.8216	**0.6579**	0.6888	0.8135	0.7957	0.8228	0.8114	0.7007	0.7641
	AdamW lr=1e^{-4}	0.8279	0.6568	0.6607	0.8232	**0.8236**	0.8191	**0.8141**	0.6793	0.7631
	SGD lr=0.01	0.8351	0.6454	**0.6977**	**0.8298**	0.7999	0.8235	0.7991	**0.7754**	**0.7757**
	SGD lr=0.03	0.8152	0.6000	0.6913	**0.8298**	0.8193	**0.8309**	0.8068	0.7386	0.7665
bert-double	AdamW lr=3e^{-6}	0.8186	0.6262	0.6844	0.7925	0.7680	0.8054	0.7989	0.7501	0.7555
	AdamW lr=5e^{-5}	0.8106	**0.6809**	0.6744	0.8134	**0.8173**	0.8208	0.8131	0.7097	0.7675
	AdamW lr=1e^{-4}	0.8130	0.6583	0.7075	0.8127	0.8155	0.8233	0.7915	0.6887	0.7638
	SGD lr=0.01	**0.8277**	0.6400	**0.7096**	0.7975	0.7924	0.8134	0.8107	0.7591	0.7688
	SGD lr=0.03	0.8166	0.6608	0.7039	**0.8168**	0.8095	**0.8353**	**0.8193**	**0.7689**	**0.7789**
bert-lstm	AdamW lr=3e^{-6}	0.8207	0.6051	0.7031	0.7809	0.7824	0.8227	0.7985	0.7343	0.7560
	AdamW lr=5e^{-5}	0.8022	0.6681	0.6811	0.8239	0.7949	**0.8436**	0.8044	0.7284	0.7683
	AdamW lr=1e^{-4}	0.8233	0.6303	**0.7105**	0.8190	**0.8129**	0.8268	0.7950	0.7429	0.7701
	SGD lr=0.01	**0.8259**	0.6524	0.6947	0.8133	0.8078	0.8108	0.8066	**0.7505**	0.7702
	SGD lr=0.03	0.8202	**0.6803**	0.7070	**0.8271**	0.8029	0.8269	**0.8124**	0.7468	**0.7780**

Table 3: QWK scores for different model architectures and hyperparameter configuration on ASAP dataset. The default lhs-cls output is used for all models.

MODEL	POOL	lhs	2lhs	3lhs	4lhs	AVE
bert-finetune	**cls**	0.8152	0.8084	0.8243	**0.8234**	0.8178
	att	**0.8398**	0.8155	**0.8377**	0.8071	**0.8250**
	mean	0.8205	**0.8358**	0.8261	0.8154	0.8245
	max	0.8333	0.8111	0.7673	0.8163	0.8070
	mm	0.8240	0.8208	0.8177	0.7490	0.8029
	conv	0.8015	0.7856	0.7825	0.7906	0.7901
bert-double	**cls**	0.8202	0.8154	0.8027	0.8259	0.8160
	att	0.8231	0.8176	0.8123	0.8272	0.8201
	mean	0.8238	0.8157	0.8182	0.8239	0.8204
	max	0.8351	0.8358	0.8215	**0.8274**	0.8299
	mm	0.8206	**0.8419**	**0.8376**	0.8169	0.8293
	conv	**0.8381**	0.8380	0.8239	0.8263	**0.8316**
bert-lstm	**cls**	0.8202	0.7983	0.8186	0.8273	0.8161
	att	0.8183	0.8137	**0.8302**	0.8083	0.8176
	mean	0.8349	0.8263	0.8089	0.8280	0.8245
	max	0.8349	**0.8444**	**0.8301**	0.8169	0.8316
	mm	**0.8446**	0.8309	**0.8302**	**0.8361**	**0.8354**
	conv	0.8321	0.8103	0.8084	0.8299	0.8202

Table 5: QWK scores using different pooling strategies on each of the last 4 layers. Best values for each model and layer are in bold.

using modified model architectures and other pooling methods over the default BERT fine-tuning.

MODEL	POOL	gl4	gf8	ahs
bert-finetune	**mean-hs**	0.8118	0.8174	0.8127
	cc	0.8127	**0.8323**	0.8164
	wl	**0.8207**	0.8270	**0.8361**
bert-double	**mean-hs**	0.8288	0.8163	0.8311
	cc	**0.8445**	0.8281	**0.8388**
	wl	0.8297	**0.8385**	0.8253
bert-lstm	**mean-hs**	**0.8265**	0.8035	0.8063
	cc	0.8158	-	-
	wl	0.8190	**0.8340**	**0.8257**

Table 6: QWK scores for different pooling strategies on different layer combinations. The best values for each model and layer combination are in bold.

We now discuss the results using different pooling strategies for a combination of multiple layers, shown in Table 6. The empty values are due to insufficient memory. For the bert-finetune model, wl pooling obtained good results, especially when we pool all 12 BERT layers. This shows that all layers contain information about the essay that can still help with the scoring performance. Concatenating the first 8 layers also gave good QWK scores, showing that the first layers can still contribute to obtaining representations relevant to essay scoring. We believe this is due to the architecture of BERT that learns more general features in the first layers that contribute to the essay score, as opposed to just the features learned at the last layers that are more specialized for language modeling.

In bert-double, the cc and wl pooling methods obtained good results. In particular, cc pooling for the last 4 layers or even all the layers achieved good performance. For combining the first 8 layers, wl pooling is the best pooling method. For the bert-lstm model, wl pooling seems to work the best. However, when it comes to pooling the last 4 layers, mean-hs pooling performed better than cc or wl. Overall, we observe the best QWK results using gl4-cc and gf8-wl, thus, we consider their combination for the next experiments.

We observe bert-double to be the best at capturing relevant information from combinations of layers, as it has consistently good QWK scores. Its performance boost over the regular bert-finetune model can be seen not just in Table 6, but also in Table 5. We attribute this improvement to the ReLU activation function between the 2 linear layers, which is known to overcome the vanishing gradients problem encountered in fine-tuning (Mosbach et al., 2020) with its close-to-linear behavior. ReLU also produces sparse representations, thus accelerating representation learning.

Based on the findings from previous results, we utilize intermediate layers using the chosen multi-layer pooling strategies and the bert-double model. We consider using a hybrid pooling method (wlcc) of using cc pooling on the last 4 layers and wl pooling on the first 8 layers so all 12 layers are utilized. We denote the output for this as ahs-cls-wlcc, which uses cls embedding for the single-layer pooling. We also consider the other single-layer pooling methods (att, mean, max, mm, conv) for the last 4 layers. We evaluate the performance using the different representations on all the essay sets and report the results in Table 7.

Compared to the default lhs-cls output, all other model outputs have improved based on the average QWK performance on all essay sets. This further shows that utilizing intermediate BERT layers strongly boosts scor-

OUTPUT	SET 1	SET 2	SET 3	SET 4	SET 5	SET 6	SET 7	SET 8	AVE
lhs-cls	0.8277	0.6400	0.7096	0.7975	0.7924	0.8134	0.8107	0.7591	0.7688
gl4-cc	**0.8445**	0.6488	0.7054	0.8099	0.8053	0.8096	0.8088	0.7567	0.7736
gf8-wl	0.8385	0.6779	0.7040	0.8030	0.8009	0.8143	**0.8194**	0.7069	0.7706
ahs-cls-wlcc	0.8293	0.6737	0.7036	**0.8316**	0.7936	0.8309	0.8040	0.7638	0.7822
ahs-att-wlcc	0.8351	0.6746	0.7168	0.8131	0.8092	0.8059	0.8089	**0.7865**	0.7813
ahs-mean-wlcc	0.8392	0.6783	0.6958	0.8186	0.8041	**0.8380**	0.8081	0.7549	0.7796
ahs-max-wlcc	0.8338	0.6884	0.7080	0.8214	**0.8320**	0.8277	0.8172	0.7714	**0.7875**
ahs-mm-wlcc	0.8406	0.6618	**0.7318**	0.8106	0.8098	0.8240	0.8120	0.7658	0.7820
ahs-conv-wlcc	0.8168	**0.7288**	0.6945	0.8097	0.8166	0.8024	0.8100	0.7545	0.7791

Table 7: QWK scores of our model using different outputs on ASAP dataset.
The best values for each essay set are shown in bold.

MODEL	SET 1	SET 2	SET 3	SET 4	SET 5	SET 6	SET 7	SET 8	AVE
EASE (SVR)	0.781	0.621	0.63	0.749	0.782	0.771	0.727	0.534	0.699
EASE (BLRR)	0.761	0.606	0.621	0.742	0.784	0.775	0.73	0.617	0.705
LSTM	0.775	0.687	0.683	0.795	0.818	0.813	0.805	0.594	0.746
LSTM + CNN	0.821	0.688	0.694	0.805	0.807	0.819	0.808	0.644	0.76
BERT	0.792	0.679	0.715	0.8	0.805	0.805	0.785	0.595	0.748
XLNet	0.776	0.68	0.692	0.806	0.783	0.793	0.786	0.628	0.743
BERT Ensemble	0.802	0.672	0.708	0.815	0.806	0.814	0.804	0.597	0.752
XLNet Ensemble	0.804	0.685	0.7009	0.795	0.799	0.805	0.8	0.597	0.748
BERT + XLNet Ensemble	0.807	0.696	0.703	0.819	0.808	0.815	0.806	0.604	0.757
HISK and -SVR	0.836	0.724	0.677	0.821	0.83	0.828	0.801	0.726	0.78
BOSWE and -SVR	0.788	0.689	0.667	0.809	0.824	0.824	0.766	0.679	0.756
HISK+BOSWE and -SVR	**0.845**	0.729	0.684	0.829	0.833	0.83	0.804	0.729	0.784
Parameter-Efficient Transformer	0.743	0.674	0.718	**0.884**	0.834	0.842	0.819	0.744	0.785
HA-LSTM+SST+DAT	0.836	**0.73**	**0.732**	0.822	0.835	0.832	0.821	0.718	0.79
BERT+SST+DAT	0.824	0.699	0.726	0.859	0.822	0.828	**0.84**	0.726	0.791
R$^{2}BERT$	0.817	0.719	0.698	0.845	**0.841**	**0.847**	0.839	0.744	**0.794**
BERT-ahs-cls-wlcc (ours)	0.8293	0.6737	0.7036	0.8316	0.7936	0.8309	0.8040	0.7638	0.7822
BERT-ahs-wm-wlcc (ours)	0.8338	0.6884	0.7080	0.8214	0.8320	0.8277	0.8172	**0.7714**	**0.7875***

Table 8: QWK scores of our chosen models and other baseline models on the ASAP dataset.
Best values for each essay set are in bold. Models that outperform ours use self-supervised tasks, multi-loss learning functions, or more intricate architectures.

ing performance by providing better essay representations. We then compare the wlcc hybrid pooling to just using cc and wl on the last 4 and first 8 layers, respectively. From this, it can be seen that all the models that use hybrid pooling have an improvement in average QWK scores, indicating that using all 12 layers contributes more relevant essay information than only using a subset of layers. This also shows that the wlcc hybrid pooling is effective.

By comparing the average QWK score obtained from using gl4-cc and gf8-wl, we can see that gl4-cc is slightly better, which suggests that concatenating the last 4 layers can give a slightly better representation for all the essay sets in general. For the most part, scores obtained by each output for the essay sets are similar, except for essay sets 1, 2, and 8. These are the essay sets that exceed the maximum token length capacity of BERT from Table 1, so the performance difference can be due to the essay length. We can also see the large average QWK difference in essay set 8, which has the largest average token length (more than double the capacity). Here, gl4-cc performed significantly better, which likely means that the more complicated features encoded in the last 4 layers contribute more to the overall quality of the essay than the more general features in the first layers.

Out of all the models using the hybrid wlcc pooling, the one that got the best average QWK score is ahs-max-wlcc. Even on the longer essay sets (1, 2, 8), it was able to give generally high QWK scores. Max pooling summarizes the primary or most activated feature in the essay that so it must be able to properly extract the features contribute most to the essay score. Moreover, since only the main features are extracted, overfitting is somehow reduced, thereby contributing to its good overall performance. Using the default cls embedding for each layer (ahs-cls-wlcc) obtained the next best performance, with the other pooling methods unable to outperform it.

4.2.2. Comparison with Baseline Models

Table 8 shows the experimental results of our models with the ahs-max-wlcc and ahs-cls-wlcc outputs and other model baselines. In terms of average QWK score, our models outperform most of the existing models except the models that use self-supervised tasks (HA+LSTM+SST+DAT, BERT+SST+DAT) or multi-loss function (R^2BERT) for the BERT-ahs-wm-wlcc model. Considering that our model architecture is very simple (BERT followed by 2 linear layers), with no ensembles, self-supervised tasks, or multi-loss functions, our model performed very well for the essay scoring task. This shows that utilizing all BERT layers is indeed effective since it captures good representations. Moreover, we can observe the great potential of our model to further improve if we optimize it further or use other strategies for improvement, such as self-supervised tasks and multi-task learning.

Our models, particularly the BERT-ahs-wm-wlcc model, were able to outperform EASE, the RNNs and hierarchical models, and the vanilla and ensemble transformer models (BERT and XLNet) on all essay sets by large margins, showing that good representation learning can go a long way. It is, however, unable to outperform the state-of-the-art statistical HISK/BOSWE models on essay sets 1 and 2, which are argumentative essays. These models are well-designed to capture the explicit structure and the semantic features that are relevant in argumentative essays. Our best model obtained the best QWK on essay set 8, showing its potential to handle very long essays and a wider score range. Although it was not able to achieve the best scores, considering our simple model architecture using only a different BERT output, it gave a consistently high performance, ranking second when it comes to the best average QWK score.

5. Conclusions

In this paper, we tackle the AES task and go beyond the traditional way of fine-tuning BERT by making modifications to the default fine-tuning network and finding the best essay representation. To find good essay representations, we utilize intermediate layers and different ways of pooling each single layer and multiple layers for the fine-tuning of BERT. We found that we can improve essay scoring performance by using all or even several BERT layers and not just the default last layer output since there is relevant essay information captured by the different layers. Aside from the canonical fine-tuning process of adding a single output layer, we also explored 2 other architectures and found that both modifications can improve performance. In addition, with some hyperparameter tuning, we were able to find good base hyperparameter configurations and eventually found that the SGD optimizer generalizes better for different essay sets than Adam optimizers using BERT. We believe our proposed simple model modification using a hybrid pooled BERT representation has great potential for the AES task and potentially long-text tasks in general.

Our study can be expanded by exploring more complicated architectures, multitask learning with other loss functions or NLP tasks, and various optimization strategies. A detailed error analysis can also be performed to detect future improvements. We also hope to evaluate our model on trait-specific scoring and cross-prompt settings. Furthermore, conducting assessments of our model across various datasets and other long-text tasks to evaluate its generalizability represents another avenue for future research.

References

Majdi Beseiso and Saleh Alzahrani. 2020. An empirical analysis of bert embedding for automated essay scoring.

Yue Cao, Hanqi Jin, Xiaojun Wan, and Zhiwei Yu. 2020. Domain-adaptive neural automated essay scoring. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1011–1020.

Mădălina Cozma, Andrei M Butnaru, and Radu Tudor Ionescu. 2018. Automated essay scoring with string kernels and word embeddings. *arXiv preprint arXiv:1804.07954*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Fei Dong and Yue Zhang. 2016. Automatic features for essay scoring–an empirical study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1072–1077.

Fei Dong, Yue Zhang, and Jie Yang. 2017. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 153–162.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2020. Investigating learning dynamics of bert fine-tuning. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 87–92.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

Nitish Shirish Keskar and Richard Socher. 2017. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*.

Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 2021. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436–444.

Nelson F Liu, Matt Gardner, Yonatan Belinkov, Matthew E Peters, and Noah A Smith. 2019. Linguistic knowledge and transferability of contextual representations. *arXiv preprint arXiv:1903.08855*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Elijah Mayfield and Alan W Black. 2020. Should you fine-tune bert for automated essay scoring? In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 151–162.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. *arXiv:2006.04884*.

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)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018. Dissecting contextual word embeddings: Architecture and representation. *arXiv preprint arXiv:1808.08949*.

Matthew E Peters, Sebastian Ruder, and Noah A Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*.

Peter Phandi, Kian Ming A Chai, and Hwee Tou Ng. 2015. Flexible domain adaptation for automated essay scoring using correlated linear regression. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 431–439.

Pedro Uria Rodriguez, Amir Jafari, and Christopher M Ormerod. 2019. Language models and automated essay scoring. *arXiv:1909.09482*.

Angad Sethi and Kavinder Singh. 2022. Natural language processing based automated essay scoring with parameter-efficient transformer approach. In *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, pages 749–756. IEEE.

Youwei Song, Jiahai Wang, Zhiwei Liang, Zhiyue Liu, and Tao Jiang. 2020. Utilizing bert intermediate layers for aspect based sentiment analysis and natural language inference. *arXiv preprint arXiv:2002.04815*.

Jingbo Sun, Tianbao Song, Jihua Song, and Weiming Peng. 2022. Improving automated essay scoring by prompt prediction and matching. *Entropy*, 24(9):1206.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1882–1891.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Yongjie Wang, Chuan Wang, Ruobing Li, and Hui Lin. 2022. On the use of bert for automated essay scoring: Joint learning of multi-scale essay representation. *arXiv:2205.03835*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv:1910.03771*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Junjie Yang and Hai Zhao. 2019. Deepening hidden representations from pre-trained language models for natural language understanding. *ArXiv abs/1911.01940*.

Ruosong Yang, Jiannong Cao, Zhiyuan Wen, Youzheng Wu, and Xiaodong He. 2020. Enhancing automated essay scoring performance via fine-tuning pre-trained language models with combination of regression and ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1560–1569.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven Chu Hong Hoi, et al. 2020. Towards theoretically understanding why sgd generalizes better than adam in deep learning. *Advances in Neural Information Processing Systems*, 33:21285–21296.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.