Combining Shallow and Deep Representations for Text-Pair Classification

Vincent Nguyen[1,2] **Sarvnaz Karimi**[1] **Zhenchang Xing**[2]
[1]CSIRO Data61, Sydney, Australia
[2]The Australian National University, Canberra, Australia
{firstname.lastname}@csiro.au
{zhenchang.xing}@anu.edu.au

Abstract

Text-pair classification is the task of determining the class relationship between two sentences. It is embedded in several tasks such as paraphrase identification and duplicate question detection. Contemporary methods use fine-tuned transformer encoder semantic representations of the classification token in the text-pair sequence from the transformer's final layer for class prediction. However, research has shown that earlier parts of the network learn shallow features, such as syntax and structure, which existing methods do not directly exploit. We propose a novel convolution-based decoder for transformer-based architecture that maximizes the use of encoder hidden features for text-pair classification. Our model exploits hidden representations within transformer-based architecture. It outperforms a transformer encoder baseline on average by 50% (relative F1-score) on six datasets from the medical, software engineering, and open-domains. Our work shows that transformer-based models can improve text-pair classification by modifying the fine-tuning step to exploit shallow features while improving model generalization, with only a slight reduction in efficiency.[1]

1 Introduction

Text-pair classification determines the class relationship between two sentences; for example, it determines the inference class relationship (entailment, contradiction, or neutral) between a premise and a hypothesis (Bowman et al., 2015). Such classification requires interpreting the semantic content of sentences to determine their relationships. Applications of text-pair classification, which we experiment with here, are *natural language inference* (Bowman et al., 2015), *question answering candidate ranking* (Ben Abacha et al., 2019) and *duplicate question detection* (Wang et al., 2019a; Yang et al., 2019). For these tasks, we consider open, biomedical, and software engineering domains.

Contemporary methods in text-pair classification use transformer encoder networks, such as BERT (Devlin et al., 2019), which are popular in natural language processing (Wang et al., 2019b; Sun et al., 2019; Zhu et al., 2019). For these encoder models, there are several studies improving the model in different aspects. Liu et al. (2019b) and Yang et al. (2019) introduce improvements to the pretraining by removing Next Sentence Prediction (NSP) and using a larger batch-size with the LAMB optimizer (You et al., 2020). SciBERT (Beltagy et al., 2019) pretrains BERT over publications from SemanticScholar and adjusts the model's vocabulary to be domain-specific. ALBERT (Lan et al., 2020) increases model depth, adds layer parameter sharing, and includes the sentence coherence over NSP.

These studies primarily improve the transformer architecture by scaling up overall model capacity through dataset source (Lee et al., 2019; Alsentzer et al., 2019) and adjustment of the pretraining task (Lan et al., 2020; Joshi et al., 2019). They use only the *classification token* as the primary feature for classification. These improvements also require pretraining the architecture, which requires a high amount of computing resources.

Increasing model size and complexity is not the only approach to improve transformer-based models. Recent research shows that these models capture different levels of information at different layers in the network (Tenney et al., 2019). The information encoded in these levels are surface-level, structural and syntactic in the lower layers, and semantic in the upper layers (Jawahar et al., 2019). Shallow features are potentially useful because interpreting the semantic content of sentences may be difficult without additional knowledge. For instance, in software engineering

[1]Code and dataset will be released upon publication.

question-answering, structural knowledge and program syntax may aid in recognizing jargon in a programming question or the grammar of a program; a program does not work if the syntax is incorrect (e.g., func() vs. func{} only differ by bracketing in the *context* of the programming language). Likewise, in medical queries, the structural and syntactic information helps recognition of medical charts; for instance, blood pressure is written as BP 80/50 and understanding this structure and syntax (numbering order) aids in the semantic interpretation of *hypotension*.

We investigate the value of these structural, syntactic and semantic features in existing pretrained models for text-pair classification tasks in the medical and software engineering fields. Our model adjusts the decoder to use the *hidden representations* of the BERT transformer encoder model by combining shallow and deep feature representations of the input sentences. As our method involves adjusting the decoder, there is no requirement to pretrain the network, allowing it to be extended to all transformer-based encoders. Our main contributions are summarized as follows:

1. We design a decoder architecture to exploit shallow and deep representations from a transformer encoder-based architecture inspired by previous research on the learning capacity in BERT (Jawahar et al., 2019; Tenney et al., 2019). Our convolution-based decoder complements the parallel computation within transformers and ensures that computing features earlier in the network does not sacrifice overall efficiency.

2. We explore multi-gradient propagation in transformer architectures to adapt features from earlier layers in the network for more direct use in the downstream task. This propagation also improves generalization on tasks with fewer high-quality training samples.

3. We evaluate and analyze our methodology on natural language inference, question entailment and duplicate question detection task that use text-pair classification. Our experiments are in three domains: medical, software engineering, and open-domain. The diversity of domains tests the generalizability of our methods.

4. We *automatically* create and release a balanced duplicate question detection dataset of 1.6 million English question pairs from StackOverflow.

2 Related Work

Text-pair classification is a specialization of text classification. Early studies on text-pair classification used rule-based inference from a knowledge base of patterns and templates for textual entailment (Dagan and Glickman, 2004). This was superseded by supervised probabilistic models such as support vector machine (Malakasiotis and Androutsopoulos, 2007), naïve bayes, and decision trees (Newman et al., 2006), as well as unsupervised algorithms such as k-Nearest Neighbours (Inkpen et al., 2006).

Since 2014 (Kim, 2014), neural network-based techniques dominated the field. They are based on Convolutional Neural Network (CNN)-based encoders (Mou et al., 2016; Yin et al., 2016), Long Short-Term Memory (LSTM) and Recurrent Neural Network (RNN) sentence interaction encoder models (Liu and Huang, 2016; Lan and Xu, 2018a,b) which use shallow reasoning over sentences to capture semantics. These methods are limited by their model's receptive fields, as they cannot model deep semantic contextual knowledge and do not directly capture shallow information (Devlin et al., 2019).

Transformer-based methods The above mentioned limitations are contrasted by transformer-based models (Devlin et al., 2019). Although not strictly designed for text-pair classification, these models can take in pair sentence inputs for classification as well as a variety of other tasks (Wang et al., 2019a).

These models have subsequently seen a rise in the medical (Ben Abacha et al., 2019; Alsentzer et al., 2019; Lee et al., 2019) and software engineering fields (Zafar et al., 2019; Tabassum et al., 2020). In medical text-pair classification, there are model ensembles (Zhu et al., 2019; Ben Abacha et al., 2019) which exploit knowledge from multi-task learning. However, these models are computationally expensive, using several multi-task deep neural networks (Liu et al., 2019a) and SciBERT models for a single prediction, which prevents their use in applications where continual retraining is required. Several improvements to the transformer model in the medical domain involved using double transfer learning, where the model is further pretrained on the target domain before fine-tuning.

BlueBERT(Peng et al., 2019) and BioBERT (Lee et al., 2019) used additional pretraining data from PubMed whereas ClinicalBERT (Alsentzer et al., 2019) used a clinical dataset MIMIC-III (Johnson et al., 2016) for pretraining data. We note that these do not change the underlying architecture of encoders, and instead improved the model's effectiveness on downstream tasks via additional pretraining on the target domain or increasing ensemble size. Our work explores the feasibility of utilizing the underlying architecture instead.

CNN-based methods Our work is inspired by previous studies in *text classification* and *computer vision*. From text classification, Very Deep Convolutional Neural Network (VDCNN) (Conneau et al., 2017) is a deep convolutional neural network for text classification. VDCNN shows the importance of shortcut connections used heavily in Residual Networks (He et al., 2016) for effective gradient propagation when training deep networks, and k-max pooling for selecting the strongest signals for classification.

In computer vision, GoogleNet (Szegedy et al., 2015) introduced auxiliary multi-gradients during training to solve the vanishing gradient problem (Hochreiter, 1998). Solving the vanishing gradient problem is important as early layers in deep neural networks contain information that correlates more strongly with the input sequence. However, these layers receive less information from gradient propagation as the gradient is propagated from the output layer to the rest of the network—where each non-linearity the gradient passed through caused a sharp reduction in its magnitude. Auxiliary classifiers are added to the intermediate layers for increased gradient propagation to the early and intermediate layers while constraining the network to utilize early and intermediate features for image recognition. We experiment with auxiliary classifiers in our work.

3 Deeply Interconnected Convolutional Transformer Network

Transformer encoders learn different features of a language at different layers (Jawahar et al., 2019; Tenney et al., 2019). We explore if combining shallow features alongside deep features would improve the effectiveness of representation learning. We therefore design a new method that exploits all the hidden features within the transformer encoder. We use CNN-based decoders connected to each layer of the transformer encoder as a low-parameter fully-connected network (Lin et al., 2014) to combine hidden features in a highly parallelized manner. In doing so, multiple gradient flows (Szegedy et al., 2015) are subsequently introduced, allowing gradient propagation to shallower network layers and aiding the learning of downstream language features in earlier parts of the network.

However, the efficacy of multiple gradient flow is determined by convergence, which poses a significant challenge for our method as we use randomly initialized decoders together with a pretrained decoder. These two models expect different input distributions. To tackle this problem, we adopt convolutional components to help in dimensionality reduction and use larger batch-sizes with the LAMB optimizer (You et al., 2020) and One Cycle Policy (Smith, 2018) for hyper-convergence. We also include residual connections to ensure a stable gradient throughout the network.

We use convolutional components over LSTMs as the recurrent step is non-parallelizable (Vaswani et al., 2017) and slowdown the parallel computation in the transformer. We adopt two configurations similar to VDCNN and GoogleNet to test the generality of our method. We demonstrate that a stronger capability of learning is enabled by our method in text-pair classification tasks, especially for domains that require structural and syntactic knowledge such as the medical and software engineering domains.

3.1 Convolutional Transformer Encoder

Our first proposed network is a Convolutional transformer Encoder(TE_{conv}), where each encoder hidden layer is connected to a residual block in the decoder (Figure 1). We base this approach on past research, which shows that BERT learns surface-level, syntactic, and semantic features, but at different layers (Jawahar et al., 2019; Tenney et al., 2019). Thus, combining the final semantic output with earlier representations could aid downstream tasks. A possible approach is to concatenate all hidden states together. However, this approach is intractable at higher sequence lengths and dimensions, which causes overfitting. Another approach is to use a linear combination, scalar mix, or simple averaging (Tenney et al., 2019; Peters et al., 2018). However, this approach loses information from the summation (e.g., it may add to zero) which reduces generalization. Instead, as an intermediary, we use

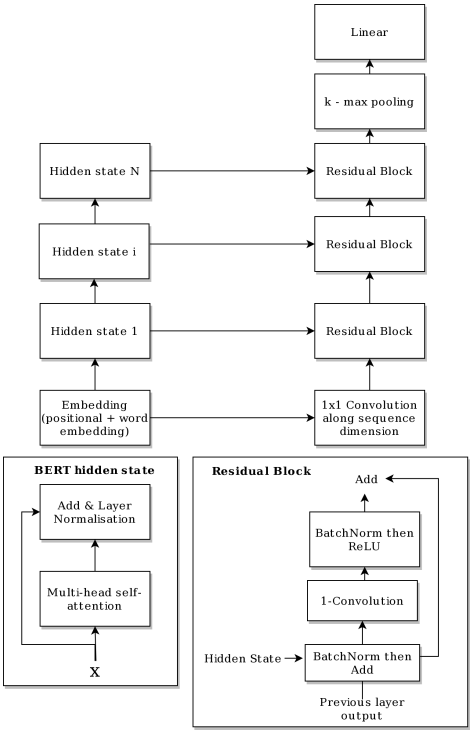

Figure 1: Network architecture of the Convolutional BERT Model.

1-convolutional filters as *information transformation gates* to transform *hidden representations*, H_i, and add it to the next hidden representation, H_{i+1}:

$$O_{i+1} = R_B(H_i) + H_{i+1}, \tag{1}$$

where R_B denotes a Residual Block, and O_{i+1} is the output for layer $i+1$ in the decoder network. Here, a residual block (Conneau et al., 2017) consists of a single 1-convolution operation along the sequence dimension. This 1-convolution acts as a fully connected linear layer across all channels with very few additional parameters (Lin et al., 2014).

Additionally, due to the use of ReLU activation function and convolutional operations, output of the CNN network must be non-negative. This output bound is in contrast with the transformer Encoder network with no bounds on the outputs. However, upon inspection, we find that the values were positive and negative and close to zero. To alleviate some of the distribution mismatch, we use batch normalization (Ioffe and Szegedy, 2015) on the hidden states inputted into the residual blocks.

3.2 Convolutional Transformer Encoder with Auxiliary Networks

Our second method, TE_{aux} uses auxiliary networks (Szegedy et al., 2015) to propagate gradients to different areas of the transformer network (see Figures 2 and 3).

An auxiliary network takes in J different hidden representations, $H_{i:i+j}$, from the transformer network. Each hidden representation undergoes dimensionality reduction using a 1-convolution to produce an output, C_i, with a feature dimension of $dim(H_i)/j$. These outputs, $C_{i:i+j}$, are concatenated and fed to a residual block, followed by k-max pooling and a fully connected layer. During inference, like GoogleNet, only the output of the final auxiliary network is used and the training loss function, ϕ_{final}, is given as a weighted sum of the auxiliary networks:

$$\phi_{final} = \phi_A^N + \alpha \sum_{i=1}^{N-1} (\phi_A^i), \tag{2}$$

where ϕ_A^i denotes the loss value of the i^{th} auxiliary network, N denotes the number of auxiliary networks, and α denotes the weight of loss value of the non-terminal auxiliary networks. We set $\alpha = 0.3$, as this was the value used in the original GoogleNet (Szegedy et al., 2015). We do not connect the auxiliary networks to avoid gradient explosions due to the double-counting of auxiliary losses propagating in the network.

We use k-max pooling before the fully connected layer for both networks to select a subset of the strongest k-signals from the feature maps. We do this for two reasons: (1) it drastically reduces the parameters in the linear layer for reduced computation cost; and, (2) it adds a layer of interpretability as the k-signals may be converted back to tokens. However, due to the bidirectional nature of transformers and padding of input, interpretability may be lost as strong signals could be from the padded portion of the sequence, which is not interpretable. In this case, attention flow (Abnar and Zuidema, 2020) might be better.

We do not use dropout in our models as dropout is a hyperparameter that requires careful tuning, which adds additional complexity.

We use pretrained weights from BERT small; however, with the exclusion of dropout, results may deviate from the literature (Wang and Manning, 2013). To avoid confusion, we named this BERT variant as Transformer Encoder (TE).

4 Datasets

We use two datasets from medical and software engineering because these domains may benefit from

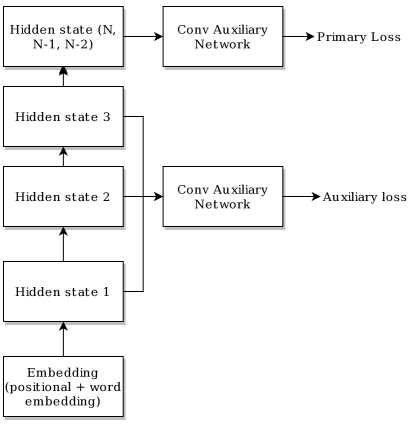

Figure 2: Network architecture of Convolutional Transformer Encoder with auxiliary networks.

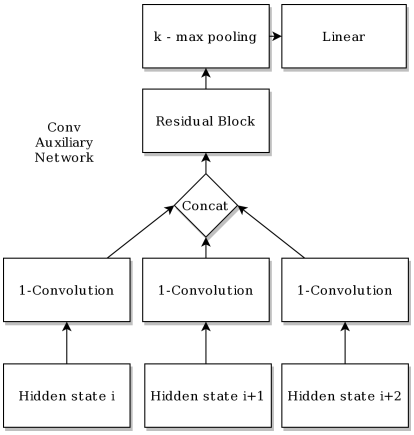

Figure 3: Architecture of the Convolutional auxiliary network.

the structural and syntactic knowledge for downstream tasks. We also use two datasets from the open-domain to test the generality of our method.

MEDIQA The MEDIQA challenge (Ben Abacha et al., 2019) was part of the BioNLP 2019 shared task. It features three separate tasks: (1) Recognizing Question Entailment (RQE), requiring binary entailment classification between text-pairs for 8,588 medical questions; (2) MEDical Natural Language (MEDNLI), a multi-label classification between premises and hypotheses for 14,049 clinical text-pairs; and, (3) Question Answering involving binary relevance classification and re-ranking between a query and retrieved answer for 476 medical questions. These datasets are smaller than those found in the open-domain because obtaining open medical data is difficult due to ethical, legal, and monetary concerns (Pampari et al., 2018; Nguyen, 2019; Ive et al., 2020). We use 5-fold cross-validation to generate non-overlapping training, validation, and testing splits as such, our results are not directly compared to the state-of-the-art (SOTA).

Original Question Conversion Error setting value for 'null Converter' - Why do I need a Converter in JSF?
Duplicate Question selectOneMenu with complex objects, is a converter necessary?
Negative Sample Conversion Error setting value '1' for 'null Converter'

Figure 4: Stack Overflow dataset examples.

Stack Overflow Dataset To test our methods on a substantial, technical dataset, we create one in the Software Engineering field. Our Stack Overflow Duplicate Question dataset evaluates the performance of our methods and its ability to generalize on specialized technical domains. To create the dataset, we use Okapi BM25 scoring (Robertson et al., 1994) from ElasticSearch with default parameters and word embeddings. Specifically, we use question titles, which we expand with word embeddings trained on the Stack Overflow corpus (Efstathiou et al., 2018) for querying. For each word in the query, we found the three most similar words in the embedding space via cosine distance and added this to the original query, Q, as expansion terms, E. To promote diversity, we empirically set the weights of the E in to be 1.3 (multiplicative), which is higher than Q at 1.0. These expanded queries, (Q, E), were used to select candidates with the highest BM25 scores not already marked as a duplicate of Q. An example from the dataset is shown in Figure 4. This dataset consists of 1.6 million question pairs, with a balanced label distribution. To our knowledge, this is the first dataset created from StackOverflow with difficult examples for text-pair classification before this work.

Open-domain We also benchmark our model against two open-domain datasets: (1) the Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) benchmark containing a collection of 570,000 text-pairs, a multi-label inference classification task; and, (2) Quora (Wang et al., 2019a), a duplicate question detection dataset of 404,000 text-pairs.

5 Experimental Setup

We use FastAI (Howard and Gugger, 2020), a PyTorch-based library. We use a one cycle policy learning rate scheduler over a cycle length of

		Method	A	P	R	F1
Open-Domain	SNLI	TE	0.798	0.599	0.609	0.604
		TE_{conv}	**0.869**†	**0.653**‡	**0.657**†	**0.657**
		TE_{aux}	0.830	0.624	0.633	0.627
		SOTA	0.923 (Liu et al., 2019b)			
	Quora	TE	0.811	0.739	0.755	0.747
		TE_{conv}	**0.880**‡	**0.842**‡	0.832‡	0.836‡
		TE_{aux}	0.879‡	0.811‡	**0.878**‡	**0.843**‡
		SOTA	0.923 (Yang et al., 2019)			
MediQA	NLI	TE	0.335	0.112	0.333	0.170
		TE_{conv}	**0.797**‡	**0.797**‡	**0.797**‡	**0.797**‡
		TE_{aux}	0.728‡	0.761‡	0.727‡	0.723‡
		SOTA	0.980 (Ben Abacha et al., 2019)			
	RQE	TE	0.557	0.278	0.500	0.358
		TE_{conv}	0.536	0.567‡	0.535	0.490‡
		TE_{aux}	**0.911**†	**0.9416**‡	**0.925**‡	**0.908**‡
		SOTA	0.749 (Ben Abacha et al., 2019)			
	QA	TE	0.575	0.287	0.500	0.365
		TE_{conv}	0.718	0.714‡	0.713‡	0.709‡
		TE_{aux}	**0.947**‡	**0.944**‡	**0.947**‡	**0.945**‡
		SOTA	0.783 (Ben Abacha et al., 2019)			
Stack Overflow	DQD	TE	0.919	0.929	0.907	0.918
		TE_{conv}	**0.943**	**0.960**	**0.926**	**0.942**
		TE_{aux}	0.939	0.952	0.924	0.938‡
Average		TE	0.667	0.502	0.592	0.529
		TE_{conv}	0.779	0.743	0.729	0.724
		TE_{aux}	**0.846**	**0.809**	**0.810**	**0.798**

Table 1: Accuracy (A), Precision (P), Recall (R) and F1-Score of different datasets. Note: † denotes a statistical significance of $p < 0.05$ and ‡ for $p < 0.001$.

15 epochs with a Label Smoothing Cross-Entropy loss and the LAMB optimizer. The peak learning rate was found using the learning rate exploration tool in FastAI. We use a batch size of 255 and a maximum sequence length of 64 for all tasks. For all other settings, we use defaults from the PyTorch transformer library (Wolf et al., 2019). We choose the best model over the 15 epochs based on the validation accuracy for test set inference. For reproducibility, we use the same seed for each experiment.

We train our models on a single GPU, V100 Tesla 16 GB. Training is repeated five times for each configuration to collect reliable statistics for paired t-test significance testing.

6 Results and Discussion

Effectiveness of the model under different dataset constraints We compare our model in differing constraints. We select the domain type (open or closed), and dataset size as constraints. We first discuss the main results.

On the SNLI dataset, a large open-domain dataset, the TE_{conv} model performs the best across all metrics. A similar observation is made for the stack overflow dataset, a large technical domain dataset, where the TE_{conv} model performs the best. On both datasets, the TE_{conv} model with auxiliary networks also performs better than the baseline. This suggests that the model performs well in data-rich environments.

This result contrasts with results from the smaller specialized datasets such as the MediQA collection. In these datasets, we see that the TE model overfits on all three medical tasks; this is apparent as the training and validation loss is lower on the TE model than the convolutional-based models. This means the model failed to generalize as test set performance was low despite performing well on the validation/training sets. This performance may be from vocabulary disparity between the train/test sets and the additional gradients which allowed convolutional models to reach a better optimum through regularization (Szegedy et al., 2015). On the NLI dataset, TE_{conv} significantly improves over TE and is stronger than TE_{aux}, suggesting that usage of all twelve layers of TE is useful for inference tasks. By contrast, TE_{aux} performed better on RQE and QA as TE_{conv} overfit on RQE and performed worse on QA.

Finally, on the Quora dataset, both TE_{conv} models performed significantly better ($p < 0.05$) with the TE_{aux} performing better in recall and F1-score while TE_{conv} performed better in terms of accuracy and precision. We note that our models do not match the state-of-the-art performance due to lack of large model ensembling and much lower total parameter count (Yang et al., 2019; Ben Abacha et al., 2019; Zhu et al., 2019) as our study is focused on investigating the usefulness of shallow features. However, our framework could potentially be applied to the current state-of-the-art models to improve their performance.

To summarize, the TE model performs poorly in the low-resource technical setting as reflected by the MediQA dataset results, MediQA has fewer training examples compared to the open-domain which made it challenging to train due to overfitting. However, we found that increased gradients allowed for better generalization. Therefore, in this low-resource setting, the convolutional TE model seems more suitable. Additionally, in more data-rich environments such as the Quora, Stack Overflow and SNLI datasets, we found the models performed better. However, on the Stack Overflow dataset, where the model performs better than the baseline, it was not statistically significant due to large variance between runs ($\sigma = 0.02$ for F1-

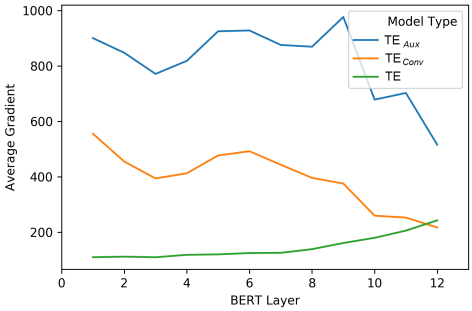

Figure 5: Average gradient over three epochs for all models on the SNLI dataset.

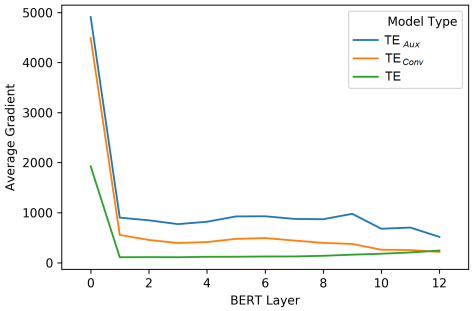

Figure 6: Average gradient over three epochs for all models on the SNLI dataset with Embedding Layer.

score).

Does including additional features from the lower layers of the network help prediction? For each domain, we select a dataset, and we analyze text-pairs to see situations where TE_{conv} benefits from additional features over the baseline. From Table 2, on the StackOverflow dataset, we see benefits from understanding using structural features and program syntax to differentiate between programming languages (e.g., angular and PHP). Similarly, on the MedNLI dataset, the model better understands medical numerical chart structure and syntax which guided better semantic understanding; which transformer encoders have been known to struggle with (Nguyen et al., 2019). Open-domain interpretation is more difficult, for instance, on the SNLI dataset, it seems additional training gradient, rather than shallow features, helped the model to learn co-reference resolution. Co-reference resolution may have helped guide the semantic understanding between sentence pairs, as this task is typically learned in deeper layers of the network (Tenney et al., 2019).

Furthermore, the model allows for improved gradient flow in the network, as shown in Figure 5, where the average gradient over three epochs on the SNLI is depicted. To illustrate each layer, we average all the gradients at that particular level. Specifically, the query, value, key attention weights gradients' are all averaged together in each hidden encoder layer. For the TE network, there is a diminishing gradient flow, a downwards slope, throughout the network. This slope contrasts with the TE_{conv} networks, which show a general increase in gradient flow (positive slope) throughout the network, allowing for learning in shallower layers of the network; which is useful because the shallower

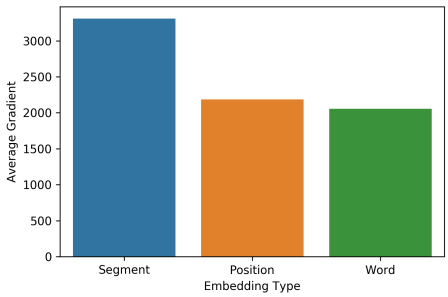

Figure 7: Average gradient over three epochs for all models on the SNLI dataset for the embedding Layer.

features, which are more correlated with the input, are now used for final layer prediction.

As the models exhibited similar trends in the embedding layer, we report a singular figure (Figure 7). A side effect of removing the NSP layer, we observe a large gradient flow in the segment(ation) embeddings, as the model learned sentence segmentation between the two text-pairs. This may explain the capacity of the network to better differentiate between the text-pairs during classification. The magnitude of this segmentation gradient is larger in both TE_{conv} models than the TE model (Figure 5), allowing for better modeling of pair semantics.

Encoder comparisons: efficiency and effectiveness Earlier, we hypothesize that using convolutional components should not hinder training speed. From Table 3, we found that TE_{conv} being a larger model, sharing the same number of layers as the original TE model, increases the training time by 13.8%, while the TE_{aux} with only three layers increases only by 7.44% in training time. TE_{aux} offers a trade-off between effectiveness and training time.

Our results indicate that both models are significantly better than the TE baseline in most settings.

Dataset	Sentence A	Sentence B	Gold Label	Baseline Prediction
StackOverflow	why does this setTimeout() call work in the console but not as a greasemonkey script?	setinterval() and .click() in a Greasemonkey script	Duplicate	Not Duplicate
	how to resolve Error: [$rootScope:inprog] http://errors.angularjs.org/1.5.8/$rootScope/inprog?p0=%24digest in dhtmlxTree	php parse/syntax errors; and how to solve them?	Not Duplicate	Duplicate
MedNLI	In the ED, initial VS were: 8 98 64 131/113	in the ED initial respiratory rate was low	Entailment	Not Entailment
	Received ASA 325mg and Nitro 0.4mg x3.	The patient has not had any vasodilator drugs.	Not Entailment	Entailment
SNLI	A young child dressed in a scarf, hat, jacket, gloves, pants, and boots, outside playing in the snow.	A child plays with a sled in the snow while dressed warmly.	Neutral	Contradiction
	A girl with a blue shirt and a girl with a striped shirt stand next to a girl with a green shirt sitting in a chair.	Two girls are standing next to a girl who is sitting.	Entailment	Contradiction

Table 2: Examples where shallow features lead to the correct prediction by our models.

The TE_{aux} model propagates higher level gradients to lower layers, takes less time to train, and is more consistent between runs as it has fewer parameters. However, the TE_{conv} model can achieve strong performance, provided the dataset is large enough.

We conduct additional experiments to verify if (1) convolutional components in the TE_{conv} network improve effectiveness; and, (2) including more layer representations (shallow and deep) to the decoder shows improvement over the baseline. These results are shown in Table 4. We conclude that the decoders can better use the encoder's features than a linear decoder from the frozen encoder experiments. Moreover, by comparing TE_{conv} with and without convolutional sub-networks, we see that convolutional components allow for better utilization of the additional features in the encoder for data-rich tasks as accuracy and F1-score increases for those tasks. Residual connections and additional features (summation of hidden states+k-max pooling-convolution) benefit medical tasks, giving an average 0.05 (absolute) boost in F1-score for each task, meaning that additional shallow features still help smaller datasets. However, we find that removing all additional parameters and utilizing only additional features provides an increase in effectiveness over the baseline. Our results are consistent with (Dong et al., 2021) which shows that skip connections are important for transformer model effectiveness. Our models exploit skip connections to combine shallow and deep representations.

Model	Total Training Time (hours)	% Slower
TE	16.75	–
TE_{conv}	19.50	13.8
TE_{aux}	18.25	7.44

Table 3: Comparison of average training time in hours between the models over a total of 15 epochs on the Stack Overflow dataset.

Model	A	F1
$\text{TE}^{\text{frozen}}$	0.583	0.505
$\text{TE}_{\text{conv}}{}^{\text{frozen}}$	0.608	0.571
$\text{TE}_{\text{aux}}{}^{\text{frozen}}$	0.584	0.544
TE_{conv} w/o Residual Block (no additional parameters)	0.700	0.668
TE_{conv}	0.711	0.636
TE	0.676	0.544

Table 4: Ablation comparisons between transformer encoders. Metrics are averaged over all tasks. Experiments encoder layers are frozen during training as denoted by *frozen*. We use mean pooling for the tasks as the classification feature token is not fine-tuned. We also include a baseline TE_{conv} model where additional parameters (such as convolution) are removed. The original splits for the MediQA datasets are used for these experiments and as such do not relate to experiments in Table 1.

Overall, we find that utilizing more layers in the TE architecture, and propagating gradient to multiple network layers allows for increased effectiveness and generalizability through regularization (Szegedy et al., 2015), especially on smaller specialized medical datasets.

7 Conclusions

We investigate whether using shallow hidden representations—which encode syntactic and structural information in the transformer encoder architecture—aids text-pair classification in the medical, software engineering and open-domains. To exploit these representations, we use deep convolutional neural networks as low-parameter networks to increase gradient propagation to the earlier layers of the network with a minimal decrease in efficiency. We find that including these representations, even as a simple summation over all hidden

states, leads to increased system effectiveness. Validating if this holds for other variants of transformer encoder architecture is a suitable avenue for future research.

Acknowledgements

Vincent is supported by the Australian Research Training Program and the CSIRO Research Office Postgraduate Scholarship. This work is funded by the CSIRO Precision Health Future Science Platform (FSP).

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online.

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, MN.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3606–3611, Hong Kong, China.

Asma Ben Abacha, Chaitanya Shivade, and Dina Demner-Fushman. 2019. Overview of the MEDIQA 2019 shared task on textual inference, question entailment and question answering. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 370–379, Florence, Italy.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal. Association for Computational Linguistics.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1107–1116, Valencia, Spain.

I. Dagan and Oren Glickman. 2004. Probabilistic textual entailment: Generic applied modeling of language variability. In *PASCAL Workshop on Learning Methods for Text Understanding and Mining*, Grenoble, France.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, MN.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. 2021. In *Proceedings of the 38th International Conference on Machine Learning*. [link].

V. Efstathiou, C. Chatzilenas, and D. Spinellis. 2018. Word embeddings for the software engineering domain. In *IEEE/ACM 15th International Conference on Mining Software Repositories*, pages 38–41, Gothenburg, Sweden.

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, Las Vegas, NV.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116.

Jeremy Howard and Sylvain Gugger. 2020. Fastai: A layered API for deep learning. *Information*, 11:108.

Diana Inkpen, Darren Kipp, and Vivi Nastase. 2006. Machine learning experiments for textual entailment. In *Proceedings of the second RTE Challenge*, Venice, Italy.

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, Lille, France.

Julia Ive, Natalia Viani, Joyce Kam, Lucia Yin, Somain Verma, Stephen Puntis, Rudolf N. Cardinal, Angus Roberts, Robert Stewart, and Sumithra Velupillai. 2020. Generation and evaluation of artificial mental health records for natural language processing. *Nature Partner Journal Digital Medicine*, 3(1):69.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy.

Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3(1):160035.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing

and predicting spans. *Computing Research Repository*, abs/1907.10529.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751, Doha, Qatar.

Wuwei Lan and Wei Xu. 2018a. Character-based neural networks for sentence pair modeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 157–163, New Orleans, Louisiana.

Wuwei Lan and Wei Xu. 2018b. Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3890–3902, Santa Fe, New Mexico. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *Proceedings of the 8th International Conference on Learning Representations*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Min Lin, Qiang Chen, and Shuicheng Yan. 2014. Network In Network. In *Proceedings of 2nd International Conference on Learning Representations*, Banff National Park, Canada.

Biao Liu and Minlie Huang. 2016. A sentence interaction network for modeling dependence between sentences. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 558–567, Berlin, Germany.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A robustly optimized BERT pretraining approach. *Computing Research Repository*, abs/1907.11692.

Prodromos Malakasiotis and Ion Androutsopoulos. 2007. *Learning Textual Entailment Using SVMs and String Similarity Measures*, page 42–47. Association for Computational Linguistics, USA.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 130–136, Berlin, Germany.

Eamonn Newman, Nicola Stokes, John Dunnion, and Joe Carthy. 2006. *Textual Entailment Recognition Using a Linguistically–Motivated Decision Tree Classifier*, volume 3944, pages 372–384.

Vincent Nguyen. 2019. Question answering in the biomedical domain. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 54–63, Florence, Italy.

Vincent Nguyen, Sarvnaz Karimi, and Zhenchang Xing. 2019. ANU-CSIRO at MEDIQA 2019: Question answering using deep contextual knowledge. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 478–487, Florence, Italy.

Anusri Pampari, Preethi Raghavan, Jennifer Liang, and Jian Peng. 2018. emrqa: A large corpus for question answering on electronic medical records. *Computing Research Repository*, abs/1809.00732.

Yifan Peng, Shankai Yan, and Zhiyong Lu. 2019. Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets. In *Proceedings of the Workshop on Biomedical Natural Language Processing*, pages 58–65, Florence, Italy.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *Computing Research Repository*, abs/1802.05365.

Stephen Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at trec-3. In *Proceedings of the Third Text REtrieval Conference (TREC 1994)*, Gaithersburg, MD.

Leslie N. Smith. 2018. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. *Computing Research Repository*, page arXiv:1803.09820.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–385, Minneapolis, MN.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, Boston, MA.

Jeniya Tabassum, Mounica Maddela, Wei Xu, and Alan Ritter. 2020. Code and named entity recognition in stackoverflow. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4913–4926.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Computing Research Repository*, abs/1706.03762.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *In the Proceedings of International Conference on Learning Representations*, pages 353–355, Brussels, Belgium.

Haoyu Wang, Mo Yu, Xiaoxiao Guo, Rajarshi Das, Wenhan Xiong, and Tian Gao. 2019b. Do multi-hop readers dream of reasoning chains? In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 91–97, Hong Kong, China.

Sida Wang and Christopher Manning. 2013. Fast dropout training. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 118–126, Atlanta, GA.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *Computing Research Repository*, page arXiv:1910.03771.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *Computing Research Repository*, page arXiv:1906.08237.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large Batch Optimization for Deep Learning: Training BERT in 76 minutes.

Sarim Zafar, Muhammad Zubair Malik, and Gursimran Singh Walia. 2019. Towards standardizing and improving classification of bug-fix commits. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–6.

Wei Zhu, Xiaofeng Zhou, Keqiang Wang, Xun Luo, Xiepeng Li, Yuan Ni, and Guotong Xie. 2019. PANLP at MEDIQA 2019: Pre-trained language models, transfer learning and knowledge distillation. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 380–388, Florence, Italy.