Hierarchical Transformer Networks for Long-sequence and Multiple Clinical Documents Classification

Anonymous EMNLP submission

Abstract

We present a Hierarchical Transformer Network for modeling long-term dependencies across clinical notes for the purpose of patient-level prediction. The network is equipped with three levels of Transformer-based encoders to learn progressively from words to sentences, sentences to notes, and finally notes to patients. The first level from word to sentence directly applies a pre-trained BERT model as a fully trainable component. While the second and third levels both implement a stack of transformer-based encoders, before the final patient representation is fed into a classification layer for clinical predictions. Compared to conventional BERT models, our model increases the maximum input length from 512 tokens to much longer sequences that are appropriate for modeling large numbers of clinical notes. We empirically examine different hyper-parameters to identify an optimal trade-off given computational resource limits. Our experiment results on the MIMIC-III dataset for different prediction tasks demonstrate that the proposed Hierarchical Transformer Network outperforms previous state-of-the-art models, including but not limited to BIGBIRD.

1 Introduction

Transformers have gained popularity and have achieved superior performance in many natural language processing (NLP) tasks. The scheme of Transformers entirely dispenses with convolution and recurrence, solely relying on multi-headed self-attention mechanisms and position-wise feed forward networks (Vaswani et al., 2017). Inspired by Transformers, the BERT model (Devlin et al., 2019) and its variants (Lan et al., 2019; Liu et al., 2019; Sanh et al., 2019; Joshi et al., 2020; Zaheer et al., 2020) have been solidly established as the state-of-the-art methods in numerous NLP studies. BERT-based models impose an input length constraint, which limits their applicability of processing multiple, longitudinal documents. To handle this challenge, previous efforts have proposed to split long documents (or, by extension, a sequence of documents) into small chunks and then aggregate their respective representations (Adhikari et al., 2019; Pappagari et al., 2019). However, these approaches do not consider the temporal interrelations between longitudinal sequences of (potentially many) documents, and also disregard the knowledge of hierarchical structure within the document (Yang et al., 2020). For humans, it is important to understand hierarchical and longitudinal document structure when reading a series of long documents, such as chapters in a full-length novel, legal documents, and clinical notes in patient trajectories. Similarly, to process longitudinal documents, a model should incorporate this information into its architecture.

Motivated by Hierarchical Attention Networks (Yang et al., 2016), we propose Hierarchical Transformer Networks to capture the structure inherent in longitudinal sequences of documents. Our model constructs three levels—from words to sentences, then sentences to documents, and finally documents to the prediction label—leveraging both temporal and structural interrelations. We utilize a BERT model directly at the word level, experimenting with different sized BERT models to evaluate the relative trade-off between model size and sequence length. At the sentence and document levels, we employ a Transformer-based encoder architecture first proposed in Vaswani et al. (2017). Also, we implement a time-aware adaptive segmentation at the document level to capture the real temporal relationship of notes across long time periods, while aggregating notes in short time periods.

We conduct experiments using clinical notes from MIMIC-III (Johnson et al., 2016). Due to the difficulty of training Transformers successfully (Popel and Bojar, 2018), we extensively experiment with numerous hyper-parameter settings to achieve a robust training system. We also integrate distributed training to resolve memory constraints and to incorporate longer input texts. We compare our proposed model with the state-of-the-art models for two clinical outcome predictions: in-hospital mortality and phenotype prediction. Our experimental evaluation shows that Hierarchical Transformer Networks consistently outperform other alternatives with an overall improvement of up to 21% in AUC, 51% in PRC and 46% in F1 score. Through extensive ablation studies, we show that the components of the Hierarchical Transformer Networks successfully process temporal and hierarchical information of clinical notes and effectively enhance clinical predictions.

We note that while the notion of a hierarchical network for Transformers might not be conceptually novel, the fact that it has not yet been proposed for processing long-sequence clinical notes demonstrates that there are serious challenges to such a method. The difficulties largely exist, for example, optimization failure without appropriate learning rates, convergence difficulty without valid initializations, overfitting easily on training sets without proper dropout. Our main contribution is to make the model applicable and feasible to train for long and multiple text classification, as we are not simply classifying an individual document, but rather large collections of documents longitudinally over time (i.e., one classification for all of a patient's notes). To the best of our knowledge, this is the earliest attempt to build the Hierarchical Transformer Network for modeling long and multiple clinical text classifications.

2 Related Work

2.1 Hierarchical Deep Learning Architecture

To handle long documents, previous works have applied hierarchical deep learning models that stack neural networks to draw inference at each level of the hierarchy (Zhou et al., 2016; Gao et al., 2018). Yang et al. (2016) first proposed the hierarchical attention network based on GRUs for document classification. Kowsari et al. (2017) later applied multiple deep learning architectures, including fully-connected DNN, GRU, LSTM, and CNN into a hierarchical model. More recent work, HiBERT (Zhang et al., 2019), presented a hierarchical architecture to pre-train document-level Transformer encoders with unlabeled data for extractive document summarization. These hierarchical models progressively learn a representation for long-term dependencies, which could in theory enable them to explicitly deal with longitudinal sequences of documents.

2.2 Transformer Models in Clinical Domain

With the wide implementation of Transformer-based models in NLP, these have also been adapted to clinical tasks. One category of such tasks is clinical predictive modeling (Si et al., 2021). In a similar paradigm of sequence modeling with Recurrent Neural Networks, Transformers attempt to model the entire patient trajectory by encoding clinical events at each time stamp(Choi et al., 2020). One of the earliest efforts intended to develop a multi-headed attention-based model for processing multivariate clinical time series data (Song et al., 2018). Recently, BEHRT (Li et al., 2020) was built based on BERT for analyzing large-scale, sequential clinical data. Another notable domain where Transformers continue to push the frontier is clinical NLP. Many studies pre-trained BERT models with biomedical literature (Lee et al., 2020; Beltagy et al., 2019) or clinical notes (Alsentzer et al., 2019; Peng et al., 2019; Si et al., 2019) to develop the domain-specific language model, and these studies showed that such models generally outperform off-the-shelf models in varied clinical NLP tasks. However, for clinical text classification (e.g., automatic ICD coding, clinical outcome predictions) which generally requires a series of clinical notes as input, BERT does not always perform well, probably due to its restriction on computational resources and its fixed-length restriction (Li and Yu, 2020; Makarenkov and Rokach, 2020; Si and Roberts, 2020). In keeping more closely with the spirit of Transformers, our work is also built on top of Transformers with an emphasized focus on effective representation of long document sequences.

2.3 Clinical Text Classification

Unstructured notes contain important details about patient status that do not exist in the structured data of Electronic Health Records (EHR). Previous studies have developed advanced neural networks to classify clinical notes with word embeddings (Liu et al., 2018). Despite the success, context-free word embeddings fail to encode the information of a given surrounding context (Si et al., 2019). More advanced pre-trained language models show their capability to provide context-sensitive representations for clinical words (Feng et al., 2020). In this work, we integrate one of the prominent language models, BERT, as the word-level encoder of our architecture to better represent clinical words. A closer comparison to our work is FTL-Trans, which implements BERT at the word level and Bi-LSTMs at the note level (Zhang et al., 2020). To this end, we propose a Hierarchical Transformer Network architecture to encode sequences of clinical notes. This goes beyond FTL-Trans by both (1) modeling an additional level (more than one document) and (2) utilizing a full stack of Transformers in the model. We hypothesize this model can learn the contextual complexity of documents, and also leverage structural and temporal information at each level of the hierarchy.

3 Model Architecture

The proposed model architecture is illustrated in Figure 1. The model progressively constructs the representation from the word level towards the final classification level. The model at each level automatically captures the important parts with multi-headed self attention and accumulates the entire sequence with pooling into the input representation of the next level. The input length is cropped or padded to a fixed size at the word, sentence, and document levels. The final representation from the document level is fed to a fully-connected dense layer with a Sigmoid function to output the prediction probability. In the following subsections, we will introduce each model component in detail.

3.1 Word-level BERT encoders

As shown in Figure 1, at the word level, a BERT model is employed and the word-pieced tokens in a sentence are fed into the model. We implement the encoder part of the BERT model to represent the words in a sentence, and all parameters in the module are trainable. Words are preprocessed to obtain the word-pieced tokens through the preprocessing module and with the same token vocabulary list used in BERT (Devlin et al., 2019). Similar to the BERT word-level module, we keep the two special tokens [CLS] and [SEP] at the start and end of the sentence respectively. The first token of each sentence is [CLS] and its corresponding hidden state is always considered as the aggregation to represent the entire sentence. [SEP] is located at the end of the sentence and it is important in differentiating sentences. We omit the segment embeddings and keep the positional encoding. Therefore, for a given token i, the input embedding E_{i} is built by concatenating the word-pieced token embedding Tok_i, and the positional encoding vector P_{i}.

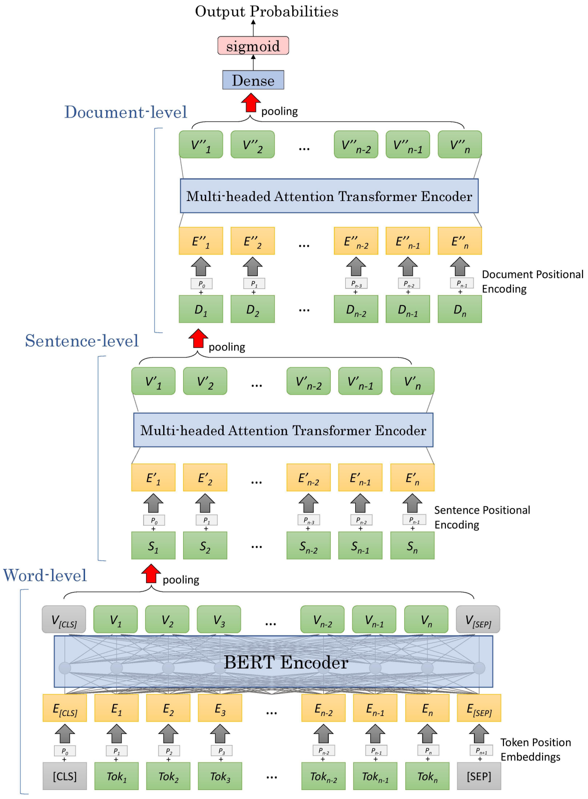

Output Probabilities
sigmoid
Dense
Document-level
pooling
V''1 V''2 ... V''n-2 V''n-1 V''n
Multi-headed Attention Transformer Encoder
E''1 E''2 ... E''n-2 E''n-1 E''n
P0 P1 Pn-3 Pn-2 Pn-1
Document Positional Encoding
D1 D2 ... Dn-2 Dn-1 Dn
Sentence-level
pooling
V'1 V'2 ... V'n-2 V'n-1 V'n
Multi-headed Attention Transformer Encoder
E'1 E'2 ... E'n-2 E'n-1 E'n
P0 P1 Pn-3 Pn-2 Pn-1
Sentence Positional Encoding
S1 S2 ... Sn-2 Sn-1 Sn
Word-level
pooling
V[CLS] V1 V2 V3 ... Vn-2 Vn-1 Vn V[SEP]
BERT Encoder
E[CLS] E1 E2 E3 ... En-2 En-1 En E[SEP]
P0 P1 P2 P3 Pn-2 Pn-1 Pn Pn+1
Token Position Embeddings
[CLS] Tok1 Tok2 Tok3 ... Tokn-2 Tokn-1 Tokn [SEP]

Figure 1: Model Architecture

3.2 Sentence- and Document-level Transformer-based Encoders

We stack Transformer-based encoders to build the representation of each sentence and each document, respectively. We briefly introduce the Transformer architecture, and for more details, we recommend the work by Vaswani et al. (2017). The Transformer-based encoder is constructed by N layers, and each layer is a residual connection of multi-headed self-attention and a fully-connected feed forward network. Each self-attention takes three inputs – Q(query), K(key), V(value) – to process through the scaled dot-product attention. The outputs from the scaled dot-product attention are concatenated and put through a linear dense layer. As opposed to a single self-attention head, Q, K, and V are partitioned into multiple heads to enable the model to attend to information at different positions from different representation subviews.

For both sentence- and document-level, positional encoding vectors are concatenated with the input states. The input state of each sentence is obtained from the first [CLS] hidden state of the respective sentence, which is termed as the *CLS-pooling* strategy. Instead, from sentence to document, and from document to label, we experiment with different pooling strategies for aggregating the representations from previous levels. This enables providing high levels of the model with more access to the lower-level representations instead of simply using what is accumulated in the `[CLS]` token. The other pooling strategies we consider consist of *mean*, *max*, and *mean_max poolings*. Take the *mean_max pooling* as an example. The average and maximum of hidden states on the sequence length axis are first obtained separately, and then concatenated to get the pooled output of the whole sequence. Following this practice, at the sentence level, the complete sequence of sentences in a given document is pooled to generate the input embeddings of the document. At the document level, the entire series of documents for a given patient is pooled to produce the corresponding patient representation. For the final label, we simply apply a dense layer with a Sigmoid function to output the classification probabilities. The model is also generalizable to be easily adapted to other machine learning NLP tasks such as pre-training, clustering, and matching, equipped with different loss functions.

3.3 Time-aware Adaptive Segmentation and Filling at Document Level

Timestamps associated with clinical notes do not always reflect the temporal reality of clinical practice. Notes often come in bursts and short real-time periods do not inherently have real temporal sequence between each other. On the other hand, notes outside a long time span contain meaningful sequential information that can be encoded by the neural network. In order to differentiate short-period co-occurrences with long-range dependencies, we dynamically merge clinical notes into groups to capture the real temporal information between notes. Meanwhile, such approaches reduce the input sequence length (i.e., number of documents) that are fed into the neural network, which enables the model to learn long-term dependencies more effectively.

For each patient, we first sort the notes in a chronological order, and then apply a greedy algorithm to find the segmentation points. The algorithm minimizes the maximum time span of contiguous groups.

Formally, given T documents in a sequence $\{d_t\}_{t=1}^{T}$, we have k-1 segmentation points $\{s_i\}_{i=1}^{k-1}$ to split the sequence into k groups $\{G_j\}_{j=1}^{k}$, where

$$G_j = \begin{cases} \{d_t \mid d_t.\,\text{time} < s_1\}\,, & \text{if j = 1} \\ \{d_t \mid d_t.\,\text{time} \geq s_{k-1}\}\,, & \text{if j = k} \\ \{d_t \mid d_t.\,\text{time} \in [s_{j-1}, s_j)\}\,, & \text{otherwise.} \end{cases}$$

where $d_t.\,\text{time}$ is the charttime of document d_t. The span of a group is defined as the time difference of the earliest and the latest document in the group:

$$\text{span}\{G_j\} = \max_{d_k \in G_j}\{d_k.\,\text{time}\} - \min_{d_{k'} \in G_j}\{d_{k'}.\,\text{time}\}$$

The optimal choice of the segmentation points can be found by minimizing the following:

$$\hat{s}_1, \dots \hat{s}_{k-1} = \underset{s_1, \dots s_{k-1}}{\text{argmin}}\{k\}$$

$$\text{subject to} \quad \max_{1 \leq j \leq k}\{\text{span}\,(G_j)\} \leq D$$

where D constrains the upper bounding of the span. Intuitively, for a given maximum time span, notes within the span are considered as one "document". The notes outside the span are segmented into different units. In this way, we attempt to preserve the temporal relationship of notes across long terms while combining the notes that come in bursts.

4 Data and Tasks

4.1 Dataset Description

Our experiments are performed with the MIMIC-III (Medical Information Mart for Intensive Care III) (Johnson et al., 2016), which is a de-identified clinical database composed of 46,520 patients with 58,976 admissions in the intensive care units (ICUs). MIMIC-III has been widely studied in clinical NLP tasks as it contains extensive resources of unstructured clinical notes (i.e., 2 million notes in the `NoteEvents` table). We describe note preprocessing in detail in Appendix A.1.

4.2 Prediction Tasks

We evaluate our proposed model to predict in-hospital mortality and phenotypes. These tasks are standard clinical outcomes of interest that are important to support clinical decisions. Note that our model is not specifically constrained to these tasks and can be extensively applied to other clinical applications. Descriptive statistics about patient cohorts are shown in Table 1.

In-hospital Mortality Prediction

MIMIC-III indicates the time of death for patients who die in the hospital, enabling us to form the cohorts for in-hospital mortality. We use `hospital_expire_flag` (in `Admissions` table) to label positive cases. In addition, to avoid confusion with multiple admissions of the same patient, we include patients with only one admission.

Table 1: Descriptive Statistics of Datasets.

		In-hospital Mortality	Phenotype Prediction
# Total Patients (% Positives)		30,881 (13.80%)	30,990 (Table A.5)
# Notes Per Patient	Mean	18.1	16.9
	Median	12	11
	80 $\%tile$	24	22
# Sentences Per Note	Mean	29.8	37.4
	Median	18	21
	80 $\%tile$	42	50
# Wordpieces Per Sentence	Mean	19.2	18.9
	Median	12	12
	80 $\%tile$	22	22
# Total Sentences		16,662,894	19,656,126
# Total Notes	Raw	906,717	866,735
	Adaptive	559,942	525,222

* $\%tile$: percentile.

We exclude *discharge summaries* in mortality prediction because discharge summaries mention the mortality outcome textually. For the same reason, we also remove all notes with charttime later than the time of death and discharge time.

Phenotype Prediction

The purpose of phenotype prediction is to classify patients into a variety of diagnoses. Specifically, we select the top **ten** relatively high-prevalence phenotypes, each of which is associated with more than 2000 patients. We consider the diagnostic ICD-9 codes to be the prediction label (a widely-used, though incomplete, surrogate for the phenotype). The phenotype disease name, ICD-9 code, disease type, and the number/percentage of patients for each phenotype in MIMIC-III are reported in Table A.5. For this task, we include all the notes up to and including the discharge date, because ICD codes are assigned after discharge.

5 Experiments

Here, we describe the compromises made in order to feasibly train such a large model on GPUs, as well as the baselines and evaluation metrics used in the experiments. Notably, Hierarchical Transformer Networks require smaller BERT models than what are normally used, even when utilizing multiple GPU architectures. To achieve a fast and effective optimization, we implement an exponential decay with linear warmup for learning rate decay.

5.1 Distributed Training

The sequence lengths required by our model are significantly longer (many thousands of words) than the standard GPU training can handle without significant compromises (i.e., the standard BERT model has a maximum input length of 512 word pieces). To resolve resource limits and augment text lengths, we implement the mirrored distribution strategy to distribute the training across multiple GPUs. We introduce the strategy with more details in Appendix A.2. Specifically, we train our proposed model on 4 NVIDIA Tesla V100 GPUs (32G), which means the batch size is quadrupled. Each training step takes approximately the same time between using 1 GPU verses using N+ GPUs, so the overall time is decreased four-fold if the training takes the same steps.

5.2 Compared Baselines

We compare the proposed model with the following alternative models:

BigBird: Zaheer et al. (2020) extend the BERT model to longer sequences with sparse attention mechanisms, which is assumed as the current state-of-the-art method for long-sequence text classification. We implement BigBird for each document at the word-level, and apply a fully-connected layer for the output probability (shown in Appendix A.3). The BigBird utilizes a flattened representation of texts, directly from word to label.

Han: The Hierarchical Attention Network (HAN) model is widely used for document classification. We follow Si and Roberts (2020) to build the architecture into a triplet structure that encodes notes over a long time (shown in Appendix A.3). The model learns the representations at each level with Bi-LSTMs and global context-based attention.

BertLstm: We also develop a variation of the proposed model, termed BertLstm, where the Transformers at the sentence and document levels are replaced with Bi-LSTMs. The architecture and model summary is shown in Appendix A.3. This allows us to measure the absolute performance improvement provided by the top-to-bottom Transformer architecture by replacing the top two Transformer levels with Bi-LSTMs layers. This model is also FTL-Trans (Zhang et al., 2020) extended to multiple documents.

To ensure a fair comparison, we enable the hierarchical models (i.e.,Han,BertLstm, and the proposed model) contain the same number of parameters (around 5.6-million), while the BigBird remains the same as in the released version (because the model is fixed). We carefully select the hyper-parameters to meet this comparison requirement. The detailed descriptions of the model hyper-parameters are described in Appendix A.3.

5.3 Evaluation Metrics

For method comparisons, we use the Area Under the Receiver Operating Characteristic curve (AUC), the Area Under Precision-Recall curve (PRC), Precision, Recall, and F1-score to report the predictive performance. The use of PRC in addition to AUC attempts to mitigate variance due to imbalanced class distributions, as the Precision-Recall curve is particularly tailored for identifying less-frequent cases. Each cohort is split into train, validation, and test, with a ratio of 8:1:1. We train the model on the train set, apply early stopping on the validation set to prevent overfitting, and report the metrics on the test set. More specifically, we calculate the loss on the validation set at the end of each epoch (a complete pass over the training data), and early stopping is triggered when the loss has been increasing for three subsequent epochs.

6 Performance Comparisons

Table 2 reports the performance comparisons of in-hospital mortality and phenotypes. We observe that our proposed model, Hierarchical Transformer Networks, outperforms other baselines for all tasks in AUC, PRC and F1-score. The performances of the flattened model, BigBird, are considerably worse than the other three hierarchical models in all tasks. This is reasonable considering the large number of notes in MIMIC-III, as the abundance of data causes the contribution from hierarchical levels to become essential.

The performances of Han and BertLstm are approximately the same. The advantages of Hierarchical Transformer Networks over BertLstm are significant in phenotype predictions with improvements of 0.0258 in AUC, 0.0541 in PRC, and 0.0542 in F1-score. And Hierarchical Transformer Networks have relatively small improvements of 0.0251 in AUC, 0.0416 in PRC, and 0.0429 in F1-score, compared to Han. This demonstrates that the Transformers applied at hierarchical levels make a steady contribution to the performance improvement. More importantly, the direct usage of BERT models at the word level has a decisive impact on the predictive performance. Note that we only adopt one layer of encoder in our proposed model, which already yields the best performance across alternatives. According to findings from the Ablation Study Section 7, the model still has room to improve by enlarging the model and incorporating more data. Thus, we believe the great potential of the Hierarchical Transformer Networks would outperform strong state-of-the-art methods in clinical outcome predictions.

Table 2: Performance comparisons in in-hospital mortality and phenotype predictions. Per-phenotype metrics are shown in Table A.6.

	Macro-AVG of 10-phenotype prediction				
	AUC	PRC	Precision	Recall	F1
BigBird	0.7497	0.4647	0.6513	0.3515	0.4421
Han	0.8845	0.6608	**0.7037**	0.5546	0.6033
BertLstm	0.8838	0.6483	0.6712	0.5733	0.5919
Our Model	**0.9096**	**0.7024**	0.7003	**0.6342**	**0.6462**
	In-hospital mortality prediction				
	AUC	PRC	Precision	Recall	F1
BigBird	0.8769	0.8139	0.6924	0.7049	0.6986
Han	0.9610	0.8992	0.7837	**0.8356**	0.8088
BertLstm	0.9608	0.8946	0.8740	0.7283	0.7945
Our Model	**0.9677**	**0.9032**	**0.8810**	0.7603	**0.8162**

*All models have the same input lengths. BertLstm and Our Model use the same BERT_{tiny} at word level.

We also note that Hierarchical Transformer Networks generate the highest PRCs in in-hospital mortality and almost all phenotype predictions (Table A.6 b). Considering the fact that PRC is a critical metric in clinical problems where properly classifying the positives is important, which is always the case in clinical outcome predictions. Higher PRC indicates that Hierarchical Transformer Network is more likely to find all the positive cases without accidentally marking negative cases as positive, and such performance is more preferred, especially in clinical phenotype predictions.

7 Ablation Study

Considerable factor of the Transformer's success relies on the right setting of hyper-parameters. We examine some of the important parameters that impact training performance, robustness, and efficiency to identify an optimal trade-off. This is critically necessary for our model as the hierarchical transformers require carefully-selected compromises to keep the model size manageable.

7.1 Input Text Lengths

The off-the-shelf BERT models are pre-trained with an input sequence length of 128, which is much longer than most sentences in clinical notes. As shown in Table 1, the number of word pieces per sentence has a mean value of around 19 (19.2 for the in-hospital mortality cohort, and 18.9 for the phenotype cohort) and a median value of 12. Thus, it might be a waste of resources to use 128 tokens at the word level. However, cutting off too many

Table 3: Performance of hypertension with different input lengths. We denote the first non-header row as the **base** input, where the models contain 80th percentile data length at the patient and document level, and 64 word pieces at the sentence level.

Sequence length at each level [Percentile]			Hypertension	
Patient	Document	Sentence	AUC	PRC
22 [80th]	50 [80th]	64 [96.7th]	0.8722	0.8327
34 [90th]			0.8720	0.8337
16 [70th]			↓ 0.8623	↓ 0.8183
	85 [90th]		0.8733	0.8299
	37 [70th]		↓ 0.8655	↓ 0.8209
		128 [98.6th]	0.8744	0.8309
		32 [90th]	↓ 0.8546	↓ 0.8147
		22 [80th]	↓↓ 0.8347	↓↓ 0.7997

*Unlisted values are identical to those of the **base** input.

Table 4: Performance of hypertension with distilled BERT models. Each BERT model is evaluated with three different settings: 1. The maximum length that the memory can afford (*Max Sequence Length*); 2. As BERT$_{base}$ incorporates only 6 documents, all the other models are fed with the same 6 documents (*Last Six Notes*); 3. Only discharge summary is fed into the model (*Discharge Summary*).

		Hypertension	
	Max Sequence Length	AUC	PRC
BERT$_{tiny}$	D50_S75_W128	0.8750	0.8181
BERT$_{mini}$	D40_S60_W64	0.8706	0.8066
BERT$_{small}$	D25_S50_W64	0.8863	0.8333
BERT$_{medium}$	D12_S50_W64	**0.8869**	**0.8365**
BERT$_{base}$	D6_S50_W64	0.8788	0.8178
	Last Six Notes		
BERT$_{tiny}$	D6_S50_W64	0.8660	0.8115
BERT$_{mini}$		0.8776	0.8213
BERT$_{small}$		0.8645	0.8040
BERT$_{medium}$		0.8763	**0.8231**
BERT$_{base}$		**0.8788**	0.8178
	Discharge Summary		
BERT$_{tiny}$	D1_S50_W64	0.8497	0.8030
BERT$_{mini}$		0.8496	0.7978
BERT$_{small}$		0.8627	0.8094
BERT$_{medium}$		0.8503	0.8036
BERT$_{base}$		**0.8649**	**0.8161**

*All other hyper-parameters are the same across all BERT models. Only the BERT models applied at the word level and the input sequence lengths are different.

tokens would also harm the pre-trained model capability. Thus, it would be interesting to evaluate such a trade-off. We evaluate the performances of **hypertension** phenotype prediction with varied input sequence lengths at different levels. The results are shown in Table 3.

We first examine the results of different sequence lengths at the sentence level, or the number of tokens in a sentence, shown in the last row in Table 3. Even though the sequence length with 128 tokens has reached to 98.6th percentile, the performance does not sizably improve (i.e., from 64 to 128, the AUC slightly increases by 0.0022). However, starting from 32, the performances *drop* steadily. For lengths of 32 and 22, they do not perform well (with AUCs of 0.85 and 0.83) although they reach the 90th and 80th percentiles, respectively. Thus, we assume that chopping off a large number of tokens out of the original 128 token input, indeed harms the pre-trained model capability.

The results with sequence lengths at the patient and document levels (i.e., the number of notes and sentences) are shown in the Patient and Document columns. We experiment with 90th, 80th, and 70th percentile data. All three settings yield an approximately comparable performance with AUC scores around 0.86 to 0.87. It is reasonable to have low performance with 70th percentile data (0.86+), but it makes a rather minor difference between 80th and 90th percentiles (0.87+).

7.2 BERT Variations

We investigate different distilled BERT models at the word level, including BERT$_{tiny}$, BERT$_{mini}$, BERT$_{small}$, BERT$_{medium}$, BERT$_{base}$ (Turc et al., 2019). The parameter sizes of the models are shown in Appendix A.4 Table A.3. Given the same memory limits, we feed into the maximum sequence length for each distilled model, and we investigate if larger models would yield better performance even with smaller input lengths. As shown in the column *Max Sequence Length* of Table 4, different models have varied max input lengths (max_seq_len:D_S_W) that can be incorporated into 4 GPU memories (128G) at maximum capacity.

Notably, the max document length for BERT$_{medium}$ is only 12, but the performance of BERT$_{medium}$ achieves the best AUC (0.8869) and PRC (0.8365) among all other combinations. For BERT$_{tiny}$, BERT$_{mini}$, and BERT$_{small}$, even though these three models incorporate many more documents than BERT$_{medium}$, the performances of them are still slightly worse than BERT$_{medium}$. Interestingly, BERT$_{base}$ performs worse than BERT$_{small}$ and BERT$_{medium}$.

Meanwhile, we investigate the impact of keeping the document length fixed at the BERT$_{base}$ max capacity of 6 documents. We run all other distilled models on the same 6 documents to evaluate if larger models would outperform smaller models given the same amount of input data. As presented in the column *Last Six Notes*, we notice that BERT$_{base}$ achieves the best AUC and BERT$_{medium}$ achieves the best PRC.

Furthermore, we evaluate our model capacity using only one document to predict the phenotype. We only process the discharge summary to predict whether the patient has hypertension. This would be more challenging than using all the notes because we have a small portion of data. We want to see if the proposed hierarchical architecture can still be used with the same architecture and achieve good performance. As reported in the *Discharge Summary* column, the models continue to perform reasonably well with AUC around 0.85. The best AUC (0.8649) and PRC (0.8161) are achieved by BERT$_{base}$.

However, compared to the performances that extensively use the majority of notes to make predictions, the results using only one note are worse. For all BERT models, the performances with the max sequence length and the last six notes outperform those only using discharge summary. Thus, we show the necessity of incorporating as many documents as possible. This is more important when the phenotype is hard to get a satisfactory performance. Adopting all possible notes into the model would yield sufficient room for improvement.

Given the results of the above experiments, along with the general mantra "more data and larger models", we conclude that sufficient data is more crucial and would further improve the performance even if the model size may not be the largest. We therefore provide an applicable recommendation for those cases with less GPU memory: we should first make sure to incorporate sufficient data, then choose the larger model.

7.3 Transformer Encoder Variations

We first evaluate the performance with different **numbers of encoder layers** ($L = 1, 2, 4, 6, 8$) in the sentence- and document-level transformers. Table 5(A) shows that the model with 2 encoder layers achieves the best AUC (0.8722) and PRC (0.8327). Notably, models with fewer layers (L=$1, 2$) generally perform better than those with more layers (L=$4, 6$). Although this is opposed to the general mantra that larger models yield better performance, we assume it is because extreme model sizes might lead to an improvement bottleneck if the model is only used as fine-tuning classification.

Table 5: Performance of hypertension predictions: (A) numbers of encoder layers, (B) pooling, (C) positional encoding, and (D) adaptive segmentation.

		Hypertension	
	L	AUC	PRC
(A)	1	0.8674	0.8218
	2	**0.8722**	**0.8327**
	4	0.8645	0.8199
	6	0.8672	0.8213
	8	0.8684	0.8285
(B)	pooling		
	first	0.8683	0.8214
	mean	0.8702	0.8295
	max	0.8675	0.8222
	mean_max	**0.8722**	**0.8327**
(C)	w/o	0.8700	0.8294
	positional encoding	**0.8722**	**0.8327**
(D)	w/o	0.8558	0.7887
	adaptive segment	**0.8722**	**0.8327**

[*] Unless specified, other hyper-parameters identical to best-performing model.

We also compare different **pooling strategies** of how to aggregate the representations from the previous to the next level. Table 5(B) finds that mean_max pooling is the best-performing pooling method.

As shown in Table 5(C), excluding **positional encodings** slightly hurts performance. Thus, position-sensitive information is necessary for each representation unit to incorporate the orders of words/sentences/documents.

The results in Table 5(D) show that there are significant decreases in AUC and PRC if we remove the **adaptive segmentation**. If clinical notes for the same patient are all independent without proper segmentation, the effect is clearly reflected in the performance (0.8558 in AUC and 0.7887 in PRC).

8 Conclusion

In this work, we develop the Hierarchical Transformer Network to effectively process the sequential and hierarchical structure of clinical notes. The model takes the interrelations among clinical notes and the multilevel hierarchical information into account. We evaluate our approach using common clinical predictions, including in-hospital mortality and phenotype predictions. Our results demonstrate that the proposed model outperforms strong baselines in AUC, PRC and F1-score for both predictions. We also perform an extensive range of experiments on the proposed model with an optimal trade-off to achieve robust and effective training given computational resource limits.

References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. DocBERT: BERT for document classification. *arXiv*, 1904.08398.

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.

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 (EMNLP-IJCNLP)*, pages 3606–3611.

Edward Choi, Zhen Xu, Yujia Li, Michael Dusenberry, Gerardo Flores, Emily Xue, and Andrew Dai. 2020. Learning the graphical structure of electronic health records with graph convolutional transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 606–613.

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 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jinyue Feng, Chantal Shaib, and Frank Rudzicz. 2020. Explainable clinical decision support from text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1478–1489, Online. Association for Computational Linguistics.

Shang Gao, Arvind Ramanathan, and Georgia Tourassi. 2018. Hierarchical convolutional attention networks for text classification. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 11–23.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, 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):1–9.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Kamran Kowsari, Donald E Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S Gerber, and Laura E Barnes. 2017. HDLTex: Hierarchical deep learning for text classification. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, pages 364–371. IEEE.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. AlBERT: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Fei Li and Hong Yu. 2020. ICD Coding from Clinical Text Using Multi-Filter Residual Convolutional Neural Network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8180–8187.

Yikuan Li, Shishir Rao, José Roberto Ayala Solares, Abdelaali Hassaine, Rema Ramakrishnan, Dexter Canoy, Yajie Zhu, Kazem Rahimi, and Gholamreza Salimi-Khorshidi. 2020. BEHRT: transformer for electronic health records. *Scientific reports*, 10(1):1–12.

Jingshu Liu, Zachariah Zhang, and Narges Razavian. 2018. Deep ehr: Chronic disease prediction using medical notes. In *Proceedings of the 3rd Machine Learning for Healthcare Conference*, volume 85 of *Proceedings of Machine Learning Research*, pages 440–464, Palo Alto, California. PMLR.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Victor Makarenkov and Lior Rokach. 2020. Lessons Learned from Applying off-the-shelf BERT: There is no SilverBullet. *arXiv preprint arXiv:2009.07238*.

Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. 2019. Hierarchical transformers for long document classification. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 838–844. IEEE.

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 18th BioNLP Workshop and Shared Task*, pages 58–65.

Martin Popel and Ondřej Bojar. 2018. Training tips for the transformer model. 110(1):43–70. Publisher: Sciendo.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Yuqi Si, Jingcheng Du, Zhao Li, Xiaoqian Jiang, Timothy Miller, Fei Wang, W. Jim Zheng, and Kirk Roberts. 2021. Deep representation learning of patient data from electronic health records (ehr): A systematic review. *Journal of Biomedical Informatics*, 115:103671.

Yuqi Si and Kirk Roberts. 2020. Patient Representation Transfer Learning from Clinical Notes based on Hierarchical Attention Network. *AMIA Summits on Translational Science Proceedings*, 2020:597.

Yuqi Si, Jingqi Wang, Hua Xu, and Kirk Roberts. 2019. Enhancing clinical concept extraction with contextual embeddings. *Journal of the American Medical Informatics Association*, 26(11):1297–1304.

Huan Song, Deepta Rajan, Jayaraman Thiagarajan, and Andreas Spanias. 2018. Attend and diagnose: Clinical time series analysis using attention models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

Liu Yang, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020. Beyond 512 Tokens: Siamese Multi-depth Transformer-based Hierarchical Encoder for Long-Form Document Matching. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1725–1734.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. In *NeurIPS*.

Dongyu Zhang, Jidapa Thadajarassiri, Cansu Sen, and Elke Rundensteiner. 2020. Time-Aware Transformer-based Network for Clinical Notes Series Prediction. In *Machine Learning for Healthcare Conference*, pages 566–588. PMLR.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document Level Pre-training of Hierarchical Bidirectional Transformers for Document Summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016. Attention-based LSTM network for cross-lingual sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 247–256.

A Appendix

A.1 Note Preprocessing

For all predictions, we keep patients more than 18 years old. We consider each note entry in `NoteEvents` as a single note. Notes labeled with `ISERROR` tags and blank entries are excluded. Notes are sorted in ascending order by `charttime`. For each patient, notes are segmented/filled according to Section 3.3. Sentence segmentation is performed simply using periods and newline characters. (It results in highly sub-optimal sentence segmentation, but this is a very challenging problem on clinical notes.) Regular expressions are applied to remove special tokens including masked Protected Health Information (PHI) and numerical digits. Even though such tokens can be matched with BERT word-pieced vocabularies, these special characters would occupy space in sentences and overall provide less meaningful information related to the clinical prediction tasks.

A.2 Mirrored Strategy

The mirrored distribution strategy is developed with data parallelism, where the same model is replicated on multiple GPU devices on a single machine and different slices of the input data are fed into them accordingly. The model variables on each GPU will be mirrored and trained independently in sync. After each epoch of training, the learned variables are aggregated across each of the GPUs using an all-reduce algorithm by NVIDIA NCCL.

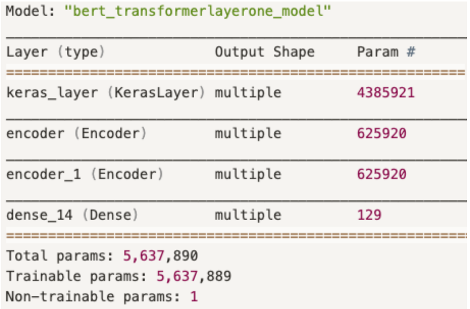
A.3 Model Hyper-parameter and Architecture

We introduce the hyper-parameter of each model in the baselines and the proposed model in this section. Note that except BigBird, we enable the compared models contain the similar number of parameters to ensure the fairness of the comparison.

Hierarchical Transformer Network: We denote L as the number of layers in the encoder, num_heads as the number of parallel heads in multi-headed attention, d_{model} as the dimension of hidden units, and d_{ff} as the dimensions of the position-wise feed forward networks. At the word level, we experiment with a series of smaller uncased BERT models with distilled knowledge including BERT$_{tiny}$, BERT$_{mini}$, BERT$_{small}$, BERT$_{medium}$, BERT$_{base}$ (Turc et al., 2019). The BERT models are downloaded from TensorFlow Hub[1] to be used as a trainable component directly. For instance, BERT$_{tiny}$ is a two-layer encoder ($L = 2$) with a 2-head self-attention ($num\_heads = 2$), and produces an output embedding with a hidden size of 128 ($d_{\text{model}} = 128$).

Table A.1: Hyper-parameter of the Hierarchial Transformer Networks

	param_name	value
word_level	num_layers	2
	d_model	128
	num_heads	2
sentence-document-levels	num_layers	1
	d_model	128
	num_heads	8
	dff	2048
	dropout	0.2



```
Model: "bert_transformerlayerone_model"
_________________________________________________________________
Layer (type)                Output Shape         Param #
=================================================================
keras_layer (KerasLayer) multiple               4385921
_________________________________________________________________
encoder (Encoder)           multiple             625920
_________________________________________________________________
encoder_1 (Encoder)         multiple             625920
_________________________________________________________________
dense_14 (Dense)            multiple             129
=================================================================
Total params: 5,637,890
Trainable params: 5,637,889
Non-trainable params: 1
```

Figure A.1: Model Summary of the Hierarchical Transformer Network with One Encoder Layer

At the sentence and document levels, we keep the encoder with the same hidden unit size as the BERT model. That is, if BERT$_{tiny}$ is used at the word level, $d_{\text{model}} = 128$ at both the sentence and document levels. We set the default values from Transformer$_{base}$ (Vaswani et al., 2017) for other hyper-parameters as follows: $num\_heads = 8$, $d_{\text{ff}} = 2048$, input position encoding dimension is the same with d_{model}, layer normalization $\varepsilon = 1e - 6$, and dropout rate $P_{\text{drop}} = 0.2$. The detailed hyper-parameter of the proposed model is shown in Table A.1.

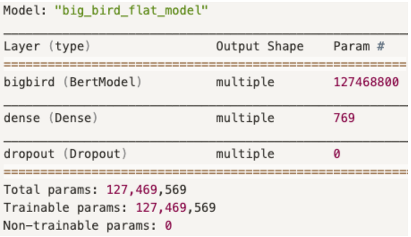
The models are trained with the Adam optimizer. More importantly, to achieve a fast and effective optimization, we implement an exponential decay with linear warmup for learning rate decay.

For the model that is specifically used in the performance comparison, we adopt an one-layer encoder both at the sentence and document levels, so that the model has around 5.6M parameters. The detailed summary of the proposed model architecture is shown in Figure A.1.

[1] https://tfhub.dev/

Table A.2: Hyper-parameter of the BIGBIRD model

param_name	value
attention_probs_dropout_prob	0.1
hidden_act	gelu
hidden_dropout_prob	0.1
hidden_size	768
initializer_range	0.02
intermediate_size	3072
max_position_embeddings	4096
num_attention_heads	12
num_hidden_layers	12
type_vocab_size	2
scope	bigbird
use_bias	TRUE
rescale_embedding	FALSE
use_gradient_checkpointing	FALSE
attention_type	block_sparse
norm_type	postnorm
block_size	16
num_rand_blocks	3
max_encoder_length	1024
vocab_size	50358



```
Model: "big_bird_flat_model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bigbird (BertModel)          multiple                  127468800
_________________________________________________________________
dense (Dense)                multiple                  769
_________________________________________________________________
dropout (Dropout)            multiple                  0
=================================================================
Total params: 127,469,569
Trainable params: 127,469,569
Non-trainable params: 0
```

Figure A.2: Model Summary of the BIGBIRD

BIGBIRD: It is a sparse-attention based transformer model that allows to handle significantly longer sequences than the original BERT model. BIGBIRD also adopts global and random attentions to a more computationally efficient attention mechanism. It shows such attentions closely resemble the full attention in BERT models. BIGBIRD also improve the performance on a wide variety of NLP tasks as a result of its capacity feeding into more input sequences. We apply the BIGBIRD for each document at the word-level. In other words, each clinical note is fed into the BIGBIRD from words. The hidden output from BIGBIRD for each note is then fed into a fully-connected network for the final classification. Although this pipeline is not the same with other compared baselines and the proposed model (flattened vs hierarchical), we assume this workflow is the current best way to implement BIGBIRD at patient-level classification (based on our preliminary experiment results). In the future, we will further investigate into how to implement BIGBIRD into a hierarchical manner.

The detailed hyper-parameter of BIGBIRD is reported in Table A.2. We also implement an exponential decay with linear warmup for the learning rate decay. The detailed model summary is shown in Figure A.2.

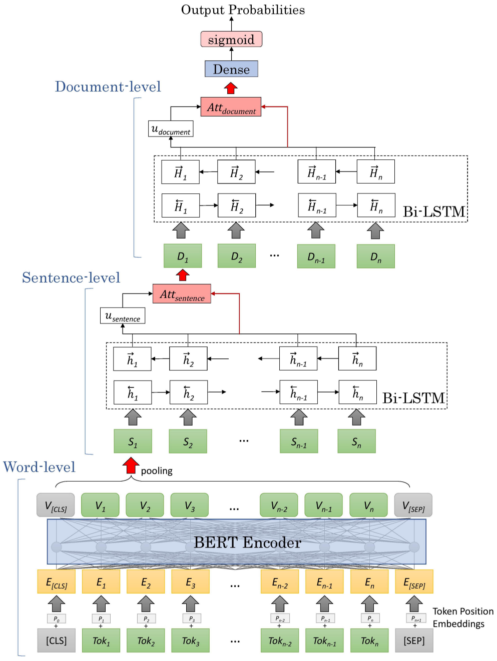
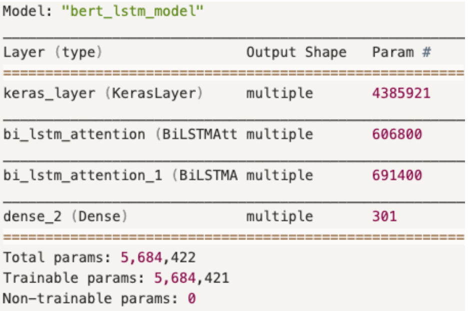

Output Probabilities
sigmoid
Dense
Document-level
$Att_{document}$
$u_{document}$
$\vec{H}_1$ $\vec{H}_2$ $\vec{H}_{n-1}$ $\vec{H}_n$
$\overleftarrow{H}_1$ $\overleftarrow{H}_2$ $\overleftarrow{H}_{n-1}$ $\overleftarrow{H}_n$
Bi-LSTM
D_1 D_2 ... D_{n-1} D_n
Sentence-level
$Att_{sentence}$
$u_{sentence}$
$\vec{h}_1$ $\vec{h}_2$ $\vec{h}_{n-1}$ $\vec{h}_n$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_{n-1}$ $\overleftarrow{h}_n$
Bi-LSTM
S_1 S_2 ... S_{n-1} S_n
Word-level
pooling
$V_{[CLS]}$ V_1 V_2 V_3 ... V_{n-2} V_{n-1} V_n $V_{[SEP]}$
BERT Encoder
$E_{[CLS]}$ E_1 E_2 E_3 ... E_{n-2} E_{n-1} E_n $E_{[SEP]}$
P_0 P_1 P_2 P_3 P_{n-2} P_{n-1} P_n P_{n+1}
Token Position Embeddings
[CLS] Tok_1 Tok_2 Tok_3 ... Tok_{n-2} Tok_{n-1} Tok_n [SEP]

Figure A.3: The BERTLSTM Model Architecture



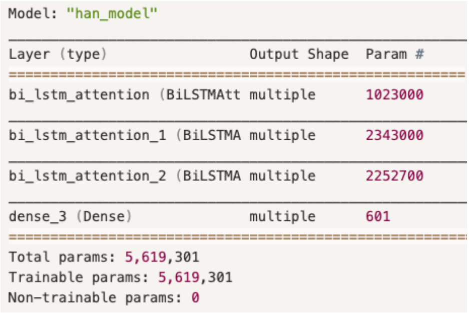
```
Model: "bert_lstm_model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer (KerasLayer)     multiple                  4385921
_________________________________________________________________
bi_lstm_attention (BiLSTMAtt multiple                  606800
_________________________________________________________________
bi_lstm_attention_1 (BiLSTMA multiple                  691400
_________________________________________________________________
dense_2 (Dense)              multiple                  301
=================================================================
Total params: 5,684,422
Trainable params: 5,684,421
Non-trainable params: 0
```

Figure A.4: Model Summary of the BERTLSTM

BERTLSTM The architecture and model summary of BERTLSTM is shown in Figure A.3 and Figure A.4, respectively. The word level still maintains a BERT model as a fully-trainable component. The sentence and document sequential information are encoded through Bi-LSTM. A global context-based attention is also adopted to capture the important knowledge and aggregate the embeddings from the previous level to the next level.

The BERT size in BERTLSTM is the same with the proposed model at the word level (BERT$_{tiny}$). The Bi-LSTM in BERTLSTM takes a hidden unit size of 200 and 150 at the sentence and document level, respectively. The output size at the document and patient level is 200 and 100, respectively.

HAN: The HAN is the same with Hierarchical Transformer Network where three layers of networks progressively build from word to sentence,



```
Model: "han_model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bi_lstm_attention (BiLSTMAtt multiple                  1023000
_________________________________________________________________
bi_lstm_attention_1 (BiLSTMA multiple                  2343000
_________________________________________________________________
bi_lstm_attention_2 (BiLSTMA multiple                  2252700
_________________________________________________________________
dense_3 (Dense)              multiple                  601
=================================================================
Total params: 5,619,301
Trainable params: 5,619,301
Non-trainable params: 0
```

Figure A.5: Model Summary of the HAN

sentence to document, and document to patient. The only difference is that we replace Transformers with Bi-LSTM for the HAN model at all layers. For Bi-LSTM in HAN, we use a hidden unit size of 300 for all three levels. The output size at the sentence, document, patient level is 300, 300, and 150, respectively. The model summary of HAN is shown in Figure A.5.

A.4 Distilled BERT Model Sizes

The model sizes with different word-level BERT models and various numbers of sentence and document transformer layers are in Table A.3.

Table A.3: Millions of parameters.

size L[1]	BERT$_{tiny}$ (4.4M)	BERT$_{mini}$ (11.2M)	BERT$_{small}$ (28.8M)	BERT$_{medium}$ (41.4M)	BERT$_{base}$ (110M)
1	5.6	13.9	35.6	48.2	121.7
2	6.8	16.7	42.4	55	133.9
4	9.2	22.2	56.1	68.7	158.3
6	11.6	27.7	69.7	82.4	182.7
8	13.9	33.3	83.4	96	207.1

[1] *L*: number of encoder layers at sentence and document level.

A.5 Parameter Allocation Experiments

We explore the effect of allocating memory to different levels of the hierarchy. to assess impact on performance. That is, given the same memory constraints and parameter sizes, we examine which level of the Hierarchical Transformer Network should be provided with more resources: the upper levels in documents and sentences, or the lower word level; and whether such allocation would impact the performance.

We train BERT$_{tiny}$_L8 and BERT$_{mini}$_L1, both of which have 13.9-million parameters. BERT$_{tiny}$_L8 allocates more to the document and sentence levels with deep encoders (L=8), but has only two layers of encoder at the word level (built in BERT$_{tiny}$). While BERT$_{mini}$_L1 allocates more to the word level with 4 layers (built in BERT$_{mini}$), but has only one layer of encoders at document and sentence levels.

Table A.4 shows training with deeper layers at the word level achieves slightly better performance than deeper layers at upper levels with the same overall model size. It indicates the hierarchical model reaches good results by focusing largely on the word layer and capturing the underlying low-level features in language, at least for phenotype classifications (perhaps other tasks may require more emphasis on higher-level representation).

Table A.4: Allocation at different hierarchical levels given the same parameter sizes. BERT$_{tiny}$_L8 represents the model applies BERT$_{tiny}$ at the word level and 8 encoder layers at the sentence and document levels. BERT$_{mini}$_L1 represents the model applies BERT$_{mini}$ at the word level and only 1 encoder layer at the sentence and document levels.

		Hypertension	
	size (M)	AUC	PRC
BERT$_{tiny}$_L8	13.9	0.8684	0.8285
BERT$_{mini}$_L1	13.9	**0.8782**	**0.8316**

A.6 Descriptive statistics of phenotype prediction cohorts

The MIMIC-III ICD-9 diagnosis table is used to determine phenotypes as the prediction labels. The detailed information about phenotypes including disease name and ICD-9 code, and the number of patients from MIMIC-III are shown in Table A.5. These are top ten of the most frequent diseases by cumulative patient counts. The selected phenotypes cover the majority of organ systems including circulatory system, genitourinary system, respiratory system, digestive system, and etc. This also indicates that our model performs well across a broad spectrum of diseases.

Table A.5: Descriptive Statistics of Phenotypes

Phenotype	**ICD-9**	**Type**	**# Patients (%)**
Essential hypertension	4019	chronic	13399 (43.2)
Coronary atherosclerosis of native coronary artery	41401	chronic	8208 (26.5)
Atrial fibrillation	42731	mixed	7525 (24.3)
Congestive heart failure	4280	mixed	6473 (20.9)
hyperlipidemia	2724	chronic	5387 (17.4)
Acute respiratory failure	51881	acute	4329 (14.0)
Pure hypercholesterolemia	2720	chronic	3874 (12.5)
Esophageal reflux	53081	chronic	3629 (11.7)
Pneumonia	486	mixed	2577 (8.3)
Chronic airway obstruction	496	chronic	2360 (7.6)

A.7 Performance of Different Models on Phenotype Prediction Tasks

We report the performance metrics in AUC (Table A.6 A), PRC (Table A.6 B), Precision (Table A.6 C), Recall (Table A.6 D), and F1 score (Table A.6 E) for all phenotype predictions using different models, shown in Table .

Table A.6: Performance metrics of Different Models for All Phenotypes

A. AUC

ICD-9	BigBird	Han	BertLstm	Our Model
4019	0.8193	0.8331	0.8693	**0.8720**
41401	0.8208	0.9587	0.9482	**0.9599**
42731	0.8023	0.9499	**0.9565**	0.9545
4280	0.7657	0.9075	**0.9216**	0.9212
2724	0.7835	0.8967	**0.9235**	0.9192
51881	0.7424	**0.9092**	0.8902	0.9083
2720	0.7461	0.8044	0.6923	**0.8693**
53081	0.7782	0.8666	0.8882	**0.8932**
486	0.6212	**0.8687**	0.8480	0.8666
496	0.6178	0.8504	0.9003	**0.9320**
Macro_AVG	0.7497	0.8845	0.8838	**0.9096**

B. PRC

ICD-9	BigBird	Han	BertLstm	Our Model
4019	0.7590	0.7817	0.8148	**0.8166**
41401	0.6967	0.9131	0.8938	**0.9163**
42731	0.6589	0.8771	0.8963	**0.8995**
4280	0.5734	0.7592	**0.7675**	0.7665
2724	0.4985	0.6940	0.7309	**0.7384**
51881	0.4068	0.6277	0.6051	**0.6396**
2720	0.4064	0.4522	0.2650	**0.5594**
53081	0.4073	0.6259	0.6532	**0.6754**
486	0.1228	**0.4131**	0.3587	0.4084
496	0.1167	0.4640	0.4976	**0.6037**
Macro_AVG	0.4647	0.6608	0.6483	**0.7024**

C. Precision

ICD-9	BigBird	Han	BertLstm	Our Model
4019	0.7187	0.7099	0.7325	**0.7625**
41401	0.8059	0.8644	0.8616	**0.8775**
42731	0.8720	0.8127	0.8456	**0.8514**
4280	0.7403	**0.8093**	0.7605	0.7833
2724	0.6580	0.6642	0.6592	**0.6667**
51881	0.6333	0.6945	**0.6950**	0.6849
2720	**0.7099**	0.6384	0.4043	0.5581
53081	0.6645	0.6779	**0.7021**	0.6467
486	0.3684	0.5797	0.5208	**0.5849**
496	0.3419	0.5864	0.5301	**0.5872**
Macro_AVG	0.6513	**0.7037**	0.6712	0.7003

D. Recall

ICD-9	BigBird	Han	BertLstm	Our Model
4019	0.6769	0.7328	**0.8155**	0.7963
41401	0.4480	0.8055	0.7559	**0.8176**
42731	0.3949	**0.8305**	0.8292	0.8238
4280	0.3713	0.5472	**0.6762**	0.5925
2724	0.3826	0.6536	**0.7875**	0.7821
51881	0.3137	0.4152	0.3913	**0.4630**
2720	0.2861	0.3165	0.1064	**0.6208**
53081	0.3169	0.5924	0.6774	**0.6979**
486	0.0279	**0.1361**	0.0850	0.1058
496	0.2964	0.5161	0.6083	**0.6419**
Macro_AVG	0.3515	0.5546	0.5733	0.6342

E. F1 score

ICD-9	BigBird	Han	BertLstm	Our Model
4019	0.6972	0.7212	0.7717	**0.7790**
41401	0.5759	0.8339	0.8053	**0.8465**
42731	0.5436	0.8215	0.8374	**0.8374**
4280	0.4945	0.6530	**0.7159**	0.6747
2724	0.4838	0.6589	0.7177	**0.7198**
51881	0.4196	0.5197	0.5007	**0.5525**
2720	0.4078	0.4232	0.1685	**0.5878**
53081	0.4292	0.6322	**0.6896**	0.6714
486	0.0519	**0.2204**	0.1462	0.1792
496	0.3175	0.5490	0.5665	**0.6133**
Macro_AVG	0.4421	0.6033	0.5919	0.6462