Clinical Prompt Learning With Frozen Language Models

Niall Taylor, Yi Zhang, Dan W. Joyce, Ziming Gao, Andrey Kormilitzin, and Alejo Nevado-Holgado

Abstract— **When the first transformer-based language models were published in the late 2010s, pretraining with general text and then fine-tuning the model on a task-specific dataset often achieved the state-of-the-art performance. However, more recent work suggests that for some tasks, directly prompting the pretrained model matches or surpasses fine-tuning in performance with few or no model parameter updates required. The use of prompts with language models for natural language processing (NLP) tasks is known as prompt learning. We investigated the viability of prompt learning on clinically meaningful decision tasks and directly compared this with more traditional fine-tuning methods. Results show that prompt learning methods were able to match or surpass the performance of traditional fine-tuning with up to 1000 times fewer trainable parameters, less training time, less training data, and lower computation resource requirements. We argue that these characteristics make prompt learning a very desirable alternative to traditional fine-tuning for clinical tasks, where the computational resources of public health providers are limited, and where data can often not be made available or not be used for fine-tuning due to patient privacy concerns. The complementary code to reproduce the experiments presented in this work can be found at https://github.com/NtaylorOX/Public_Clinical_Prompt.**

Index Terms— **Clinical decision support, few-shot learning, pretrained language models (PLMs), prompt learning.**

I. INTRODUCTION

BOTH academic and industry research works have become dominated by the use of transformer-based pretrained language models (PLMs). These are large neural networks that have been pretrained on massive amounts of openly available text data (e.g., the pile [1]). Subsequently, these are often adapted for various downstream natural language processing (NLP) tasks. These models slightly vary in their architecture, size, and language modeling objectives. Architectures are often

either encoder-only (e.g., Bidirectional Encoder Representations from Transformers (BERT) [2]), encoderdecoder (e.g., T5 [3]), or decoder-only (e.g., the GPT family [4]). In terms of parameterization and the size of the training dataset, the best-known PLMs can range from 110 M parameters and can be pretrained on approximately 3.3 B tokens (e.g., BERT [2], [5]), to 540 B parameters and trained on approximately 780 B tokens (e.g., PALM [6]). Yet, for some others, their size and architecture are kept proprietary and are unknown (e.g., GPT-4 [7]). To adapt a model to a specific application, these PLMs are often first downloaded from an online hub (e.g., from HuggingFace [8]) and further trained or fine-tuned for a given task (e.g., sequence classification by introducing additional neural network layers downstream of the base PLM) or, more recently, via forms of prompt learning.

Independently of their architecture, size, or how they are adapted to the new task, PLMs typically do not perform well on domains that have limited openly available corpora. One very important example is electronic health records (EHRs) in medical applications [9], [10], where specialized and idiosyncratic language is used and writing style (e.g., the use of domain-specific abbreviations) differs from that found in open-source corpora. For instance, BERT models pretrained on open-source general text and then adapted to specialized clinical text do not achieve the state-of-the-art performance [11], [12]. To address this limitation, many groups have developed domain-specific PLMs, which are pretrained on a large collection of biomedical and clinical text [10], [12], [13], [14], [15], [16], [17], [18]. We will refer to these as clinical PLMs. These clinical PLMs can then be adapted to a specific clinical task with more success than PLMs trained on general text.

Even with domain-specific pretraining, fine-tuning (to further adapt a clinical PLM for a downstream task) will typically require retraining all parameters of the PLM, which can be computationally expensive, time-consuming, and unsuitable for resource-constrained environments. Furthermore, fine-tuning can lead to catastrophic forgetting of the pretrained knowledge, overfitting to the smaller fine-tuning dataset, and achieve lower generalizability to samples of the new task that were not present in the fine-tuning dataset [19], [20], [21]. In the case of clinical PLMs, this lack of generalizability has been observed, for instance, when fine-tuning on American English clinical text and then validating on British English clinical text [22].

To avoid the problems of traditional fine-tuning, prompt learning has emerged as a promising alternative to adapt PLMs to downstream NLP tasks. By reformulating the downstream

task as a language modeling text-to-text prediction task [3], [23], [24], prompt learning uses the PLM without having to introduce task-specific layers on top. A few reliable approaches to prompt learning have already emerged, such as prompting few-shot examples [4], prompt tuning [25], prompting task explanations [26], p-tuning [27], chain-of-thought [28], and self-reflection [29]. These approaches have largely been tested in tasks for which there are plentiful open-source data, but they have been scarcely explored in the clinical domain. Not having to fine-tune the entire PLM is especially attractive in this domain, where data and computing resources are limited. Nevertheless, much of the prompt learning research thus far has relied upon the largest PLMs (e.g., PaLM [6] and GPT-3 [4]), which even without any fine-tuning requires massive storage and computational power (to serve a 175 B size model, it would require approximately 350 GB of vRAM and the weights of the model alone around 300 GB in memory). There has been little exploration of prompt learning approaches with smaller PLMs such as BERT.

In this study, we explore the suitability of prompt learning to adapt PLMs to clinical classification tasks. We compare prompt learning to the more traditional fine-tuning approach, e.g., using a classification head on top of the PLM. We also directly compare the performance of each approach when the PLM itself is fine-tuned or frozen during adaptation to the downstream tasks. We assess each approach in terms of its performance, training speed, required number of samples, and required number of parameters. We also assess different types of prompting, such as manual, soft, or mixed prompts. Our findings show that prompt learning can outperform traditional fine-tuning on various clinical tasks, making it a more resource-efficient and privacy-preserving alternative for clinical settings. This study provides a framework for applying prompt learning to other PLMs and downstream classification tasks, suited to resource-constrained environments such as the clinical domain.

II. RELATED WORK

Numerous studies have examined prompt learning in well-established NLP benchmarks, such as the Stanford Sentiment Treebank v2 (SST2) dataset and the General Language Understanding Evaluation (GLUE) dataset [4], [23], [24], [25], [27], [30]. A consistent finding across these studies is that prompt learning can achieve performance comparable to traditional fine-tuning when using some specific types of prompt learning, such as few-shot prompts or prompt tuning [27]. However, prompt learning appears similarly sensitive to the PLMs pretraining. One recent study used few-shot prompt learning to adapt one of the largest PLMs (GPT-3) to clinical tasks, observing a decrease in performance compared to similar tasks from the general domain [31]. This indicates that even the largest PLMs may not yield optimal results when directly applied to specialized domains, necessitating the development of domain-specific PLMs.

In a recent study, prompt learning was employed to assess the zero-shot performance on a clinical task using various clinical and nonclinical PLMs using natural language prompts [32]. The authors found that clinical PLMs outperformed general PLMs for a given task. Our work builds

TABLE I
EXPERIMENTAL SETUPS FOR ADAPTING A PLM TO DOWNSTREAM CLINICAL TASKS. PLM PARAMS ARE THE MODEL PARAMETERS OF THE BASE PLM

Approach	PLM params
PLM + Prompt learning	Frozen
PLM + Prompt learning	Fine-tuned
PLM + Classification head	Frozen
PLM + Classification head	Fine-tuned

upon these findings by introducing different prompt learning training strategies and then analyzing their performance across different clinical decision tasks. Furthermore, we investigate the impact of few-shot settings on both prompt learning and fine-tuning.

Other parameter-efficient methods to adapt a PLM to a new task include inter-layer adapters [33] and low-rank adaptation (LoRA) [34]. However, prompt learning still has the capacity to leverage expert knowledge, which is especially beneficial in domains where there are highly trained human experts, such as the medical domain. This knowledge can be leveraged by designing the prompts in collaboration with human experts, which can help build prompts that maximize the performance. Consequently, we reserve the examination of alternative methods (other than prompt-based methods) for future research endeavors.

The key contributions of this work are given as follows.
1) Multiple prompt learning strategies trained for various clinical classification tasks.
2) Direct comparison of prompt learning versus traditional fine-tuning for adapting PLM's to clinical downstream tasks.
3) We are the first to apply prompt learning directly to an EHR dataset in the form of Medical Information Mart for Intensive Care III (MIMIC-III).
4) Introduction of an International Classification of Diseases version 9 (ICD-9)-based triage task for the MIMIC-III dataset.

III. METHODS

A. Experimental Design

Throughout this article, we will be compared two approaches for adapting a PLM to a clinical downstream task. One approach is the more traditional method of combining the base PLM with a classification head on top, as shown in Fig. 1. The other approach is prompt learning, which introduced prompts and verbalizers to the PLM and does not require an additional classification head, as shown in Figs. 2 and 3. For both approaches, the PLM base can remain frozen or be fully fine-tuned (all parameters updated) during the task adaptation. The four key experimental setups are shown in Table I.

B. PLM With a Classification Head

A PLM can be adapted to a new task, such as sequence classification, by introducing additional layers on top of the PLM in the form of a classification head (traditional fine-tuning). In our implementation, the PLM is fed with an input sample x, which consists of a string of w number of words converted to tokens using the PLMs tokenizer that has a fixed
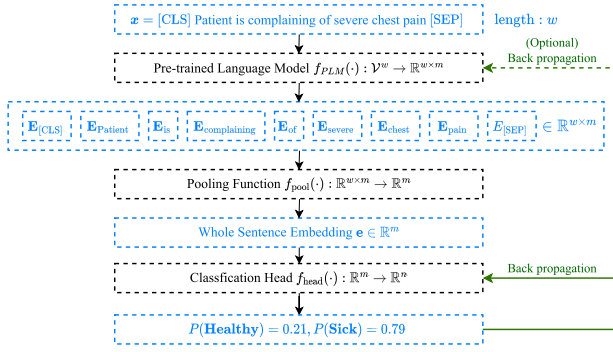
Fig. 1. Illustration of the traditional fine-tuning method with a PLM + classification head. The option to freeze the PLM is shown in dotted lines. Here, [CLS] and [SEP] tokens are special tokens for BERT-based models added to the beginning and end of sequences, respectively.

vocabulary $\mathcal{V}$. Mathematically, we say that $x \in \mathcal{V}^w$. The exact vocabulary of a PLM depends on the tokenization method used, although the BERT-based models reported upon in this work use a word-piece tokenizer [2], [14]. This PLM $f_{\text{PLM}}(\cdot)$ then transforms the input x into a list of so-called embeddings $\mathbf{E}_i$, one per token i. Each embedding is a numerical representation of the corresponding token, and it consists of a vector of m numbers

$$f_{\text{PLM}}(x) = [\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_w] \qquad (1)$$

where $f_{\text{PLM}}(\cdot): \mathcal{V}^w \to \mathbb{R}^{w \times m}$, $\mathbf{E}_i \in \mathbb{R}^m$ for all i, and $[\cdots]$ represent the concatenation of vectors.

Next, the output of the PLM is fed to a pooling function $f_{\text{pool}}(\cdot)$, which averages all the embeddings along the word dimension to obtain a single vector representation $\mathbf{e}$ of the whole input[1]

$$f_{\text{pool}}([\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_w]) = \mathbf{e} \qquad (2)$$

where $f_{\text{pool}}(\cdot): \mathbb{R}^{w \times m} \to \mathbb{R}^m$ and $\mathbf{e} \in \mathbb{R}^m$.

Finally, the output of the pooling function is fed to the classification head $f_{\text{head}}(\cdot)$, which has the task of calculating the logits y_j of each of the possible c classes $j \in \mathbb{C}$. A softmax operation is applied to the logits to produce a normalized probability score that x belongs to each of c possible classes. Mathematically, for one sample with vector representation $\mathbf{e}$, the probability of the sample belonging to class j is

$$f_{\text{head}}(\mathbf{e}) = [y_1, y_2, \ldots, y_c]$$

$$P(j) = \text{softmax}([y_1, y_2, \ldots, y_c]) = \frac{\exp(y_j)}{\sum_{k=1}^{c} \exp(y_k)}$$

where $f_{\text{head}}(\cdot): \mathbb{R}^m \to \mathbb{R}^c$ and $y_j \in \mathbb{R}$ for all j. The classification head can have any number of layers (depth) $d \in \mathbb{N}$, we opted for $d = 2$. The full architecture is represented diagrammatically in Fig. 1. Please note that in Figs. 1–3, the blue boxes refer to data, and the black boxes refer to transformation or operations. In Figs. 2 and 3, the green boxes refer to the manual prompt template, the orange box refers to the masked token position, and the purple box refers to soft or trainable prompt components.

[1] Different methods for obtaining a single vector representation of a sequence exist.
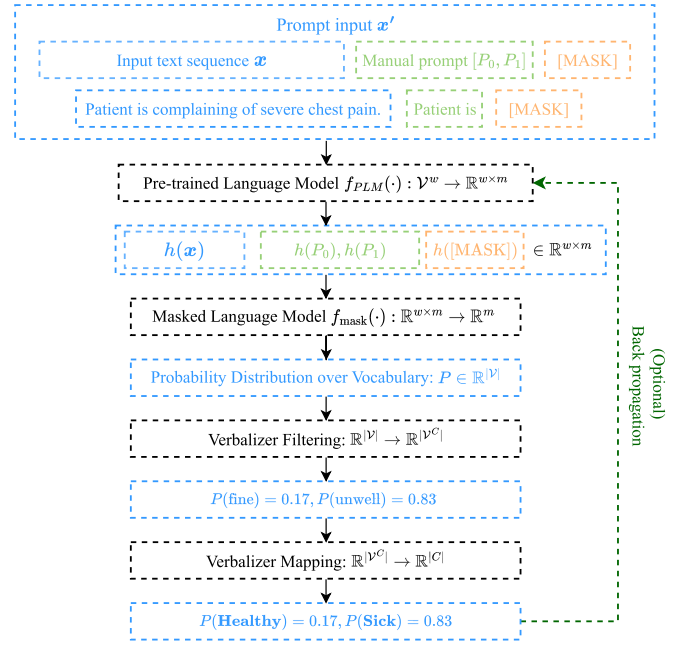

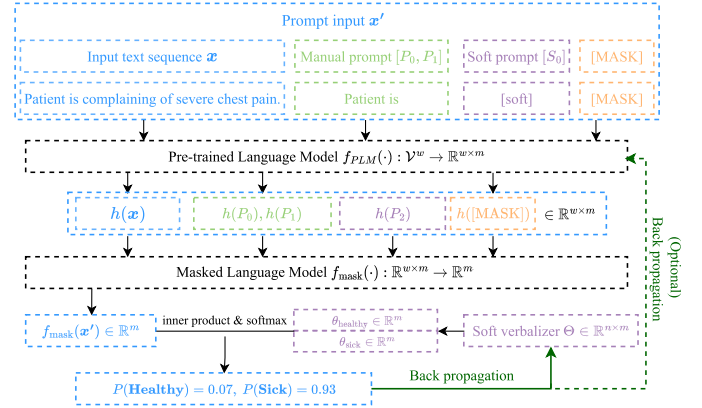Fig. 2. Illustration of the manual template and verbalizer components in the prompt learning framework.

Fig. 3. Illustration of a mixed template and soft verbalizer in the prompt learning framework. If the ⟨[soft]⟩ token S_0 is not defined manually in advance, the embedding $h(S_0) \in \mathbb{R}^m$ will be randomly initialized in the hidden space.

C. Prompt Learning

Prompt learning can be achieved via the following steps: given an input text x, we modify it to a prompt format $x' = f_p(x)$, where f_p, often called a template, will normally prepend, append, or insert a number of additional token embeddings to the original input along with a masked token, denoted by ⟨[MASK]⟩. We then feed x' into the PLM to predict the masked token, which is the same as the masked language modeling (MLM) pretraining objective of most BERT-based models. For simplicity, in this section, we use $h(x) = [\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_h] \in \mathbb{R}^{w \times m}$ to denote the embedding of an input sequence x of length w. The possible values to be predicted depend on the PLM's vocabulary $\mathcal{V}$, as defined by the tokenizer used for pretraining. A second and crucial step is to map tokens or words in the known vocabulary of the PLM to class labels in the downstream task, achieved with a

mapping $g:\mathcal{V} \mapsto \mathcal{C}$, where $\mathcal{C}$ is the set of classes. This is known as verbalization. The verbalizer can be seen as a mapping between single or multiple different tokens to distinct class labels. We denote the mapping from the hidden representation of ⟨[MASK]⟩ as $f_{\text{mask}}:\mathbb{R}^{w \times m} \to \mathbb{R}^m$, where w is the sequence length of x'. The embedding or hidden state represented at the ⟨[MASK]⟩ position output by the PLM is then passed through a feedforward classification head, and probabilities related to the derived label tokens are retrieved and used to determine the predicted token and in-effect class.

A simple prompt-based clinical classification example could be to determine whether a patient has heart disease with class labels as sick and healthy, a prompt learning setup could be as follows: take the template "⟨clinical text⟩ ⟨prompt = "Patient is"⟩ ⟨[MASK]⟩," where ⟨clinical text⟩ represents the original input text and the ⟨[MASK]⟩ token is the label or class to predict. The verbalizer will map certain tokens to each class of sick and healthy separately, essentially a dictionary mapping, e.g., {"**Healthy**": "fine," and "**Sick**": "unwell"}. Subsequently, if the token predicted at the ⟨[MASK]⟩ position is "fine," then this will be mapped to the **Healthy** class. Thus, the sentence "Patient is complaining of severe chest pain." will first be wrapped by the predefined template as "Patient is complaining of severe chest pain. Patient is ⟨[MASK]⟩." The wrapped sentence is then tokenized and fed into the PLM to predict the probability distribution over the PLM's vocabulary on the ⟨[MASK]⟩ token position, although only the probabilities of the tokens ("fine" and "unwell") that are mapped to each class that is contained in $\mathcal{V}$ are considered, and the model should have a higher probability for "unwell" and predict the class "sick." We offer an illustration of the basic prompt framework in Fig. 2.

Within the broad prompt learning framework, there are a large number of possible design configurations. For brevity, we will distinguish manual templates and verbalizers, from soft templates and verbalizers.

1) Manual Prompt Learning: We refer to the prompt learning strategy with handcrafted templates and verbalizers as manual templates and manual verbalizers, respectively. This strategy was first proposed as pattern-exploiting training (PET) [35]. A manual template has the form of $x' = \{[P_0, P_1, \ldots, P_j], x, [P_{j+1}, P_{j+2}, \ldots, P_k], [\text{MASK}]\}$, where for $i \in \{0, 1, \ldots, k\}$, P_i denotes the token of the template. As x' is fed to the PLM to get $h(x')$, the prompt tokens P_i are also mapped to the embedding space of the PLM. We denote the set of words in the verbalizer for each class $y \in \mathcal{C}$ to be $\mathcal{V}^y$, which means that each class may have multiple assigned words. The step in which the words are mapped to each class is called verbalizer filtering, denoted by $g':\mathcal{V} \mapsto \mathcal{V}^C$, where $\mathcal{V}^C$ is the set of classes with their assigned words. The probability of each class given the input x and its prompt x' is thus

$$P(y|x) = \frac{\exp\left(\frac{1}{|\mathcal{V}^y|} \sum_{t \in \mathcal{V}^y} P_M(t|x')\right)}{\sum_{i=1}^{|\mathcal{C}|} \exp\left(\frac{1}{|\mathcal{V}^i|} \sum_{t \in \mathcal{V}^i} P_M(t|x')\right)}.$$

Manual templates and verbalizers are discrete and bound to the PLMs pretrained vocabulary, so there are no additional parameters to train, although fine-tuning the PLM is still possible. The engineering of the manual prompt learning is not straightforward, with large variations in performance emerging from small changes to the tokens, and typically domain expertise is required.

2) Soft Prompt Learning: One can, however, sacrifice the human interpretability of the manual prompt learning components and use trainable or soft prompt components (similar to prompt tuning [25]). Soft prompt learning operates similar to the manual approach but replaces the fixed manual components with trainable or soft embeddings (continuous vectors) of the same dimension as the original PLM embeddings. The error from the downstream task can then be backpropagated to update only these new embeddings of the soft template and verbalizer. The soft prompt template is achieved by using trainable embeddings via the following form: $x' = \{[S_0, S_1, \ldots, S_j], x, [S_{j+1}, S_{j+2}, \ldots, S_k], [\text{MASK}]\}$, where for $i \in \{0, 1, \ldots, k\}$, S_i denotes the token of the soft template. As x' is fed to the PLM to get $h(x')$, the prompt tokens S_i are also mapped to the embedding space, where we can assume $h(S_i)$ to be optimized during training and such tokens are denoted as ⟨[soft]⟩ in the template format. Optionally, S_i can be initialized from an existing word token embedding from the PLM, such as in a manual template, or a random vector with the same dimension.

A template where all tokens are ⟨[soft]⟩ is called a soft template, while a template with a mixture of manual and ⟨[soft]⟩ tokens is called a mixed template. An illustration of the data flow for a mixed template and soft verbalizer setup is presented in Fig. 3.

Similarly, a soft verbalizer can be interpreted as replacing assigned class label words from the manual verbalizer with trainable vectors (embeddings) for each class. Therefore, when using the soft verbalizer, there is no need to build the mapping from vocabulary $\mathcal{V}$ to class labels $\mathcal{C}$ as there was for the manual verbalizer. A caveat of this is that these trainable vectors do not implicitly have semantic meaning as they were not trained through a language modeling objective. The resulting verbalizer then becomes a matrix operator $\Theta \in \mathbb{R}^{n \times m}$, where n represents the number of classes and m represents the dimension of the generated embeddings. We denote the ith row of Θ as θ_i for each trainable embedding of class i. Then, the output is processed with $f_{\text{mask}}:\mathbb{R}^{w \times m} \to \mathbb{R}^m$, where w is the sequence length of x'. Therefore, the probability of class y given the input x and its prompt x' can be calculated by

$$P(y|x) = \frac{\exp\left(\theta_y^\top f_{\text{mask}}(h(x'))\right)}{\sum_{i=1}^{n} \exp\left(\theta_i^\top f_{\text{mask}}(h(x'))\right)}.$$

D. Pretrained Language Models

To investigate prompt learning in a setting similar to the environment in which it may be deployed (e.g., in a clinical environment), we chose BioClinicalBERT [11], [14]: a clinical PLM pretrained on a large collection of PubMed abstracts and full articles [11], followed by further training on all MIMIC-III notes. PLMs have been shown to benefit from domain-specific pretraining [16]—here, we investigate whether prompt learning can utilize PLMs for downstream tasks and compare the

performance to traditional fine-tuning with a classification head. Results throughout the main body of this article will focus on BioClinicalBERT, and however, results for a selection of similarly sized clinical and nonclinical PLMs are presented in Appendix D.

E. Clinical Dataset

We used the MIMIC-III [36], a medical dataset developed by the MIT Lab for Computational Physiology. It is comprised of deidentified EHRs associated with 38 597 critical care patients and 58 976 intensive care unit (ICU) admissions at the Beth Israel Deaconess Medical Center between 2001 and 2012. Data include demographics, vital signs, laboratory tests, medications, caregiver notes, imaging reports, and mortality in and out of hospital. While the clinical tasks presented here may benefit from utilizing the multimodal data available for each patient, we focus on the use of free text clinical notes. Full details and code for reproducing these datasets and experiments are provided by authors.[2]

IV. EXPERIMENTS—CLINICAL TASKS

A. ICD-9 50

Within the MIMIC-III, data are standardized ICD-9 codes, which are used to record diagnosis and procedures. A common task is to classify the ICD-9 diagnosis code based on a patients' data for billing reasons, and it has been an objective to do so with the unstructured text data alone [37]. There are approximately 2000 unique diagnosis codes present in the MIMIC-III dataset with a very skewed distribution, resulting in an extreme multiclass problem that is beyond the scope of this article. For our classification task, we opted to subset the top 50 most frequent ICD-9 diagnosis codes that have a corresponding set of clinical notes, as has been done previously [37], [38], [39].

B. ICD-9 Triage Task

A potential concern with the ICD-9 diagnosis code classification task is that the codes themselves may be mentioned explicitly in the notes [37],[3] and furthermore, simply classifying patients' ICU discharge notes into ICD-9 diagnosis codes lacks ecological validity as a clinical decision support task. For example, within a hospital setting, patients admitted to an ICU will be treated and then "stepped down" (discharged) to another ward or team to progress their treatment when they no longer require ICU. We collaborated with clinicians to design a novel task that aims to make the classification task more similar to the decision-making process of arranging patient flow during discharge from the ICU. For example, a patient being discharged from the ICU after a cardiac event may be stepped down to a cardiology team (if the admission to ICU resulted from primary heart failure) or general internal medicine (if the cardiac event was secondary to, e.g., sepsis or multiorgan

failure). Similarly, a patient admitted to ICU with obstetric complications will likely be stepped-down to a maternity ward. In essence, we grouped together the ICD-9 diagnosis codes into "teams" that reflect the post-ICU triage destination (or "patient-flow") decision-making found in hospital settings.

For this task, we selected the top 20 most frequent ICD-9 diagnosis codes in MIMIC-III and a clinician provided triage groups based on which a team would likely continue the patient's care on being stepped down from ICU. The training classes are therefore many-to-one mappings of ICD-9 codes to "discharge destination teams" and we derived the following seven post-ICU destination teams: cardiology, obstetrics, respiratory medicine, neurology, gastroenterology, acute or internal medicine, and oncology. The resulting dataset consists of 15 000 clinical notes across the seven triage categories. For further details, see Appendix B.

C. In Hospital Mortality

A frequently used benchmark clinical support task with the MIMIC-III dataset is the prediction of whether a patient will survive their hospital episode. Within the MIMIC-III database are structured data relating to the mortality status of a patient, which, paired with a date and timestamp, allows for easy labeling of the data. Only notes prior to the mortality flag are considered, and some simple regular expression rules were used to filter any notes that had explicit mentions of a patient's death, similar to that of previous work [37], [40].

D. Length of Stay in ICU

Predicting how long a patient will require treatment in the ICU is of significant value to hospitals who aim to optimize the flow of patients in resource-limited settings (i.e., there are usually very few ICU beds compared to the hospital's overall bed capacity). We model this as a four-way classification task, binning length of stay (LoS) in the following categories: under three days, 3–7 days, 1–2 weeks, and more than two weeks [37]. These are given class labels of 0, 1, 2, and 3.

E. Full and Few-Shot Training

We will be comparing the evaluation performance of the modeling approaches in full and few-shot training setups. An important note for our few-shot experiments is that the sample size will refer to the number of samples per class for the training set, i.e., $N = s \times c$, where N is the total training set, s is the sample size per class, and c is the number of unique classes.[4] All evaluation results presented are on the full held-out test sets for each task with the different training sample sizes.

V. RESULTS

A. Comparison of Different Prompt Learning Setups

The number of possible combinations of templates and verbalizers in the prompt learning framework is vast, and

---

[2]The complementary code to reproduce experiments is provided at https://github.com/NtaylorOX/Public_Clinical_Prompt

[3]It was shown that samples where diagnosis was not mentioned explicitly only had a slight drop in performance.

[4]Note that in some instances, not all classes can fill the sample size, so for some few-shot experiments, there will remain a class imbalance.

TABLE II
COMPARING DIFFERENT PROMPT LEARNING SETUPS ON THE ICD-9 TRIAGE TASK. PLM PARAMS REFERS TO WHETHER THE PLM BODY WAS FROZEN OR FINE-TUNED DURING TRAINING. FOR PROMPT COMBINATIONS, T REPRESENTS TEMPLATE AND V REPRESENTS VERBALIZER

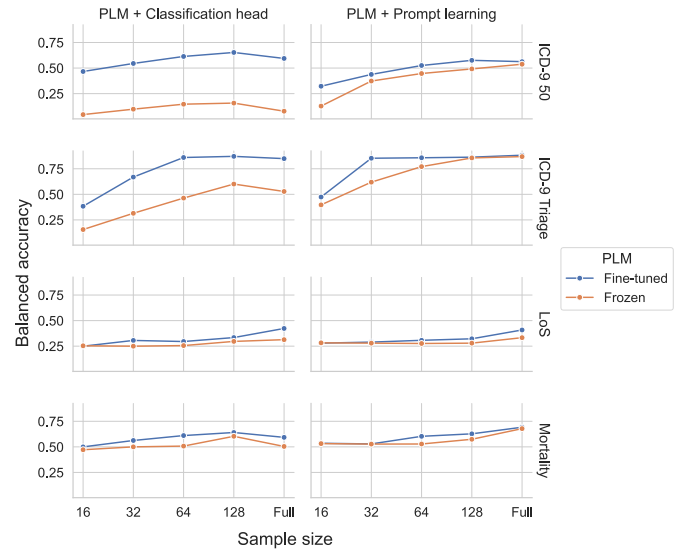| PLM params	Prompt combination(T,V)	Balanced accuracy
Fine-tuned	(manual, manual)	0.8765
	(manual, soft)	0.8818
	(mixed, manual)	0.8817
	(mixed, soft)	0.8824
	(soft, manual)	0.8860
	(soft, soft)	**0.8954**
Frozen	(manual, soft)	0.7524
	(mixed, manual)	0.8474
	(mixed, soft)	0.8724
	(soft, manual)	0.8591
	(soft, soft)	**0.8900**

Fig. 4. Balanced accuracy for training the PLM with prompt learning (left) and a classification head (right) across the four clinical tasks given by each row. The "Full" sample size refers to a full training dataset that varies in size from task to task. The color of the line refers to whether the underlying PLM was frozen or fully fine-tuned during training.

exploring all is beyond the scope of this work. As such, we have opted to utilize previous research to derive the most suitable prompt learning setups for our use case. We conducted an initial experiment comparing the performance of six prompt learning combinations on one clinical task to establish the best performing combination, which we carried through to the other clinical tasks. We chose the ICD-9 Triage task as the baseline task due to it being a relatively straightforward multiclass classification problem with a reasonably balanced distribution of classes when compared to the other tasks. The prompt learning setups were all possible combinations of a manual, mixed, or soft template with a manual or soft verbalizer. The results are summarized in Table II, where prompt combination can be read as (template, verbalizer), i.e., a manual template with a soft verbalizer is (manual, soft). Note that there are no results for a (manual, manual) prompt combination. This is because when the PLM parameters are frozen with this prompt combination, there would be no parameters to update with respect to the clinical task. As this experiment aimed to investigate how training and evaluation performance was affected by different prompt learning setups, we do not report results where no training has occurred.

The evaluation performance between the different prompt template and verbalizer combinations was very similar when the PLM was fine-tuned, but when the PLM is frozen, large differences emerged. This is linked in part to the varying number of trainable parameters introduced by each prompt learning setup (manual components do not introduce any), as well as the known limitation of prompt engineering where subtle changes to prompts can lead to performance differences [41]. While the soft template and soft verbalizer combination performed the best overall, we opted to the combination of mixed template and soft verbalizer as our prompt learning benchmark going forward. The mixed template offers a degree of human-level interpretability by allowing the injection of domain-specific prompt tokens while maintaining the power of trainable soft token embeddings alongside. For examples of mixed templates used for the tasks, see Appendix C.

B. Prompt Learning Versus Classification Head

We report balanced accuracy across different training sample sizes for the four clinical tasks for the clinical PLM with either prompt learning or a classification head in Fig. 4. Each framework utilizes the exact same PLM and the results for both fine-tuning and freezing the entire PLM are presented. In the case of the frozen PLM, only the parameters introduced by the classification head or prompt learning components are updated during training. We found that prompt learning can match or improve on using a classification head, with a much smaller gap in performance between the frozen and fine-tuned PLM setting across few-shot and full training setups.

Performance for any experimental setup on the LoS and mortality prediction is relatively poor while being in line with previous research [37]. This leaves little room for interpretation of how one training approach may enhance the task performance, with only subtle differences between the use of prompt learning and a classification head. The two tasks related to ICD-9 diagnosis codes show clear differences between the experimental setups, with prompt learning achieving a higher performance across the sample sizes when the PLM was frozen during training.

C. Training Hyperparameter Search

Prompt learning and traditional fine-tuning each have large hyperparameter spaces. Initial experiments used sensible hyperparameters based on previous research [8], [42], and using prompt learning or a classification head achieved similar performance when the PLM was fine-tuned (see Fig. 4). However, when freezing the PLMs, prompt learning appeared superior even with substantially fewer task-specific trained parameters. In order to ensure that the performance differences were not due to poor hyperparameter choices, we conducted a hyperparameter search within our single GPU's capabilities. We chose the ICD-9 Triage task as the test suite. The hyperparameter search space is provided in Table III.

The results of the subsequent optimized models for the ICD-9 Triage task are presented in Table IV. Further details

TABLE III
HYPERPARAMETER SEARCH SPACE USED FOR OPTIMIZATION OF PROMPT LEARNING AND CLASSIFICATION HEAD FINE-TUNING

Parameter	Search space
classifier learning rate	$\log.\text{uniform}[1 \times 10^{-5}, 3 \times 10^{-1}]$
batch size	$[4, 8, 16]$
gradient accumulation steps	$\text{range}[2, 10]$
dropout	$\text{range}[0.1, 0.5]$
optimizer	$\text{categorical}[\text{adamw, adafactor}]$
prompt learning rate	$\log.\text{uniform}[1 \times 10^{-5}, 3 \times 10^{-1}]$
verbalizer learning rate	$\log.\text{uniform}[1 \times 10^{-5}, 1 \times 10^{-1}]$

TABLE IV
EVALUATION METRICS FOR THE ICD9 TRIAGE TASK AFTER TRAINING WITH DERIVED OPTIMAL HYPERPARAMETERS USING PROMPT LEARNING OR A CLASSIFICATION HEAD WITH FROZEN PLMs

Approach	Balanced accuracy	F1 weighted	AUC
PLM + Classification head	0.8162	0.8919	0.9811
PLM + Prompt learning	**0.8698**	**0.9246**	**0.9889**

of the hyperparameter search and results are presented in Appendix A.

D. Sensitivity Analyses

Results suggested that on all tasks, prompt learning outperformed the traditional fine-tuning model when using a frozen PLM (see Fig. 4). A possible explanation may be the larger number of parameters (introduced by the classification head) in traditional fine-tuning leads to overfitting. To reasonably determine the effect of varying quantities of additional trainable parameters for both prompt learning and traditional fine-tuning, we varied the number of additional parameters introduced and compared the performance on the ICD-9 Triage task (see Fig. 5). Concretely, adjusting the number of trainable parameters for traditional fine-tuning involves adjusting the number of layers and hidden dimension size of the classification head. For prompt learning, adjusting number of trainable parameters requires just changing the number of soft template tokens and whether to include a soft verbalizer (manual templates and verbalizers have no trainable parameters). A training set of 128 samples per class was used as this approached peak performance without requiring a full training run. Note that prompt learning with the fewest trainable parameters (N params $= 1536$) achieves comparable performance to the traditional fine-tuning model with 1000 times the number of trainable parameters (N params $= 1\,552\,007$).

The variability in prompt learning performance based on the template and verbalizer has been well established [23], [24], [42]. We opted to focus on the use of a mixed template format, which is based around designing a common sense manual template for the task alongside soft tokens (embeddings), as shown in Fig. 3. To determine whether mixed templates benefit from a common sense or domain-specific manual template, we compared the performance of different templates, including one with a mix of unrelated and random tokens. Results are shown in Table V and we can see that having just one soft token or a set of random and unrelated manual tokens leads to a drop in performance.
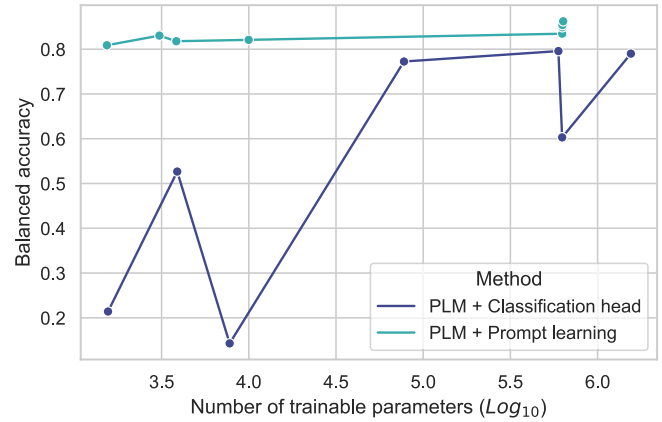

Fig. 5. Balanced accuracy for prompt learning versus traditional fine-tuning with a classification head, each with an increasing number of trainable parameters on top of frozen PLM. For readability, a logarithmic scale is used on the x-axis.

TABLE V
BALANCED ACCURACY FOR DIFFERENT MIXED PROMPT TEMPLATES FOR THE ICD-9 TRIAGE TASK

Prompt text	Balanced accuracy
<[soft]>: "This" <[MASK]>	0.8195
<[soft]>: "This" patient <[soft]>:"should go to" <[MASK]>.	0.8539
<[soft]>: "This" patient should <[soft]>:"go to" <[MASK]>.	0.8491
<[soft]>: "This" patient should <[soft]>:"go to this medical team based on symptoms of their illness" <[MASK]>.	**0.8624**
random words here <[soft]>:"random" <[MASK]>.	0.8346

TABLE VI
MEMORY STORAGE, TRAINING TIMES IN MEGA BYTES (MB), AND GPU vRAM REQUIRED FOR PARAMETERS INTRODUCED BY PROMPT LEARNING AND A CLASSIFICATION HEAD USED FOR FINE-TUNING BioClinicalBERT ON THE ICD-9 TRIAGE TASK. TRAINED PARAMS REFER TO THE SET OF MODEL PARAMETERS THAT ARE UPDATED WITH RESPECT TO THE DOWNSTREAM TASK DURING TRAINING. TRAINING TIMES ARE BASED ON ONE EPOCH WITH A BATCH SIZE OF 8

Trained params	N trained params(#)	Storage(MB)	Train time(secs)	GPU vRAM(GB)
Soft prompt	7144	0.012	83	4.6
Classification head	595,975	2.26	90	5.2
PLM + soft prompt	108.3×10^6	415.7	110	11.2
PLM + classification head	108.9×10^6	475.5	115	11.6

E. Comparing Compute and Storage Requirements

The storage requirements for the additional parameters introduced by prompt learning and the classification head are an order of magnitude smaller than for the PLM itself. When keeping the PLM frozen during training for downstream tasks, these additional parameters can be portable and remain specific to a given task while preserving a static PLM. This can enable efficient reuse of the same base model for different tasks and datasets. The comparison of the number of introduced trainable parameters for traditional fine-tuning and prompt learning with comparable evaluation performance is presented in Table VI.

VI. DISCUSSION

The experiments presented directly compare adapting a PLM to a clinical downstream task using either prompt learning or a more traditional classification head. The objective was to ascertain whether the performance for prompt learning in general domain text datasets translates to the clinical domain.

We present four clinical decision tasks related to patient outcomes in both full training and few-shot setups. Prompt learning can typically match the performance of traditional fine-tuning and even outperform traditional fine-tuning in the few-shot setting. Most notably, the performance of prompt learning when the PLM was entirely frozen was able to outperforming traditional fine-tuning with considerably fewer trainable parameters (see Fig. 5). The tradeoff between performance, efficiency in training, portability, and reusability of the underlying language model is important to consider. In resource light environments, where new data may be ingested frequently or the style of data or desired task may change regularly, then training and storing only the prompts or whichever task-specific parameters have been introduced is more desirable than retraining the entire PLM each time.

The exact reason as to why prompt learning leads to better performance through injecting trainable parameters within the PLMs input, as opposed to a classification layer on top, is not clear. One could postulate that embedding the task in a format similar to that of the original pretraining objective used for the PLM better aligns the task. Further research is required to investigate this more extensively.

A. Limitations

1) Pretraining Data Leakage: The choice of clinical PLM for the reported MIMIC-III tasks was BioClinicalBERT [14], which itself had been pretrained on MIMIC-III notes. This may in turn have made the tasks easier or inflated the evaluation performance, especially for the prompt learning approach which reframes the downstream objective to be similar to the original pretraining objective. Nevertheless, we do show that prompt learning still outperformed traditional fine-tuning with other clinical or nonclinical PLMs where data leakage is not an issue in Appendix D.

2) Task Performance Variance: We presented four clinical tasks derived from MIMIC-III notes data, and while we achieved results in line with previous research [37], the relative performance on the LoS and mortality prediction tasks was quite poor regardless of the framework. This limits the interpretability of framework differences in performance, and whether one is more suitable to some tasks than others. Similarly, we found that using a hyperparameter search for the ICD-9 Triage task improved the frozen PLM performance of the traditional fine-tuning approach by a reasonable margin and a more extensive hyperparameter search may shift this further. Nevertheless, this was also true for the prompt learning approach, but these models appeared far more robust to changes in hyperparameters and still required substantially fewer trained parameters.

B. Conclusion

Our study found that prompt learning outperformed a traditional fine-tuning approach when the PLMs are frozen prior to training on the downstream task. Prompt learning also requires fewer trainable parameters to achieve superior performance when compared to training a classifier head with a frozen PLM. The ability to utilize a single frozen PLM and share

TABLE VII
OPTIMIZED HYPERPARAMETERS FOR EACH TRAINING PARADIGM

hp	Traditional fine-tuning	Prompt learning
learning rate	0.0048	0.0121
batch size	8	4
gradient accumulation steps	4	3
dropout	0.382	0.1536
optimizer	adamw	adafactor
verbalizer learning rate	n/a	0.007

or reuse the embeddings across a number of task-specific modules, each with their own trainable prompt, is desirable for clinical applications. In the field of clinical decision support tools, a computationally efficient and interpretable model with good sufficient performance but with the facility to be used a CPU is prima-facie more desirable than a trillion-parameter model that requires high-performance computing clusters with large arrays of GPUs. The prompt learning framework is an evolving paradigm with variants being introduced regularly, and thus, we cannot claim to have covered the entire scope of prompt learning in this work. We have opted to use the most readily available and resource-efficient prompt approach to achieve our results. This work can act as a basis for further clinical prompt learning work and may encourage the use of relatively small domain-specific PLMs rather than relying on large, general, domain-independent PLMs. Future work would benefit from exploring a wider range of task and other parameter-efficient methods.

APPENDIX A
TRAINING DETAILS

We implement our experiments using a combination of the OpenPrompt framework [42], transformers, and pytorch-lightning packages. All code utilized the PyTorch framework for deep learning. For prompt learning, we use Adafactor [43] optimizer for soft and mixed templates, and AdamW [44] optimizer for language models and soft verbalizers. For traditional fine-tuning, we use AdamW optimizer for MLP heads and language models. To keep training of all setups consistent, we ran experiments using a Nvidia RTX 1080 Ti GPU. Further details of training and hyperparameters can be in the complimentary code repository. Unless otherwise stated, all evaluation metrics reported used the held-out test set and used the best performing checkpoint for each of the model training setups.

Table VII shows the derived optimal hyperparameters for each training paradigm based on the hyperparameter random search. The search consisted of 100 training runs using randomly generated hyperparameters from the search space shown in Table III. Due to relatively limited computational resource, this was only performed for the ICD-9 Triage task and a subsample of the training data was used, similar to that of our few-shot experiments with 128 samples per class.

APPENDIX B
DATASET DETAILS

A. Mortality and LoS

A combination of available clinical notes pertaining to the outcome of interest was used, including admission and
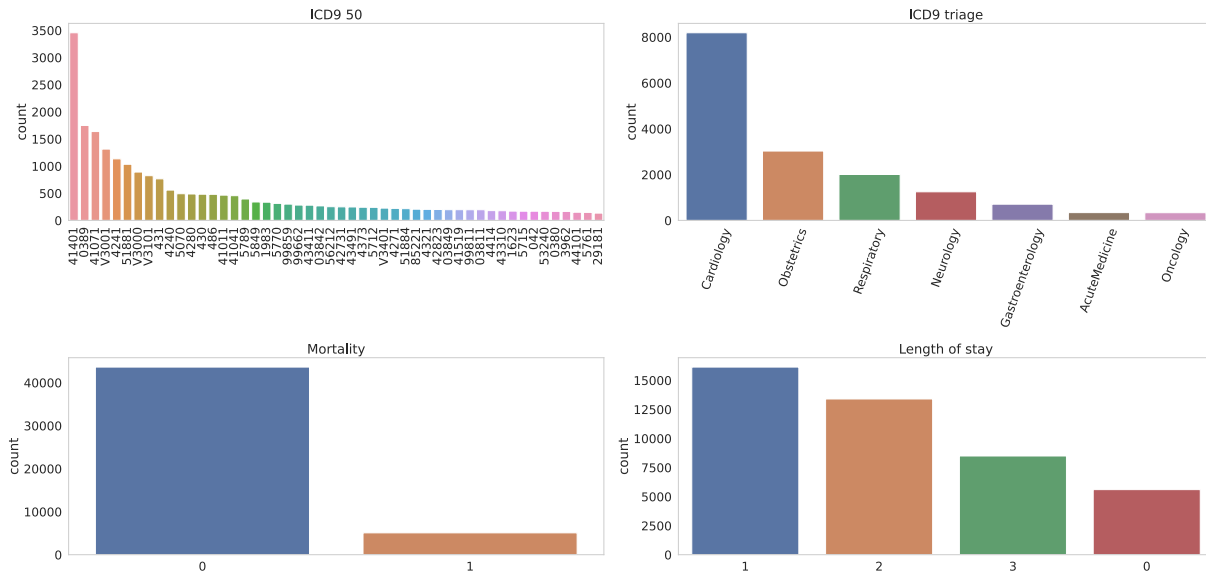
Fig. 6.   Distribution of classes for each clinical task prior to few-shot downsampling.

discharge summaries. Each task dataset was created separately and a 70–10–20 split of training–validation–test sets was used. We followed the data engineering steps outlined in the clinical outcomes paper [37].

B. ICD-9 50 and ICD-9 Triage

The ICD-9 50 task used discharge summary notes corresponding to the top 50 most frequently occurring ICD-9 diagnosis codes. The production of the ICD-9 Triage task was derived from taking the top 20 ICD-9 diagnosis codes. From this subsample, a clinician-derived suitable group represents the destination team on discharge from ICU: cardiology, obstetrics, respiratory medicine, neurology, gastroenterology, acute or internal medicine, and oncology.

Fig. 6 shows the class distributions for each of the clinical tasks presented in this article.

APPENDIX C
PROMPT EXAMPLES

Examples of different prompt methods are shown. For each task, we show one manual prompt template and one mixed template. The ⟨[soft]⟩ token represents the trainable continuous vector or embedding of the mixed template that has been initialized from the PLMs vocabulary. Thus, ⟨[soft]⟩: "This" indicates a soft embedding initialized from the PLMs pretrained embedding for the token "This."

A. ICD-9 Diagnosis Code Triage

1) ⟨clinical note⟩ The best department is ⟨[MASK]⟩.
2) ⟨clinical note⟩ ⟨[soft]⟩: "This" patient should ⟨[soft]⟩: "go to this medical team based on symptoms of their illness" ⟨[MASK]⟩.

TABLE VIII
BALANCED ACCURACY RESULTS FOR PROMPT LEARNING AND
TRADITIONAL FINE-TUNING USING MICROSOFT'S PUBMEDBERT

Sample size	Traditional fine-tuning	Prompt learning
16	0.1554	0.2249
32	0.1521	0.3749
64	0.4048	0.4621
128	0.5621	0.7814

B. Mortality Prediction

1) ⟨clinical note⟩ Patient is on the path to ⟨[MASK]⟩.
2) ⟨clinical note⟩ ⟨[soft]⟩: "This" patient ⟨[soft]⟩: "on path to" ⟨[MASK]⟩.

C. ICD-9 Diagnosis Code Classification—Top 50

1) ⟨clinical note⟩ Patient has diagnosis ⟨[MASK]⟩.
2) ⟨clinical note⟩ ⟨[soft]⟩: "This" patient ⟨[soft]⟩: "has diagnosis" ⟨[MASK]⟩.

D. LoS Prediction

1) ⟨clinical note⟩ The patient will be at hospital with a ⟨[MASK]⟩ length.
2) ⟨clinical note⟩ ⟨[soft]⟩: "This" patient ⟨[soft]⟩: "will be in hospital for a" ⟨[MASK]⟩ length.

APPENDIX D
PROMPT LEARNING VERSUS TRADITIONAL FINE-TUNING
WITH OTHER PLMS

The PLM used for the results in the main body of this article was the BioClinicalBERT [14], which has been pretrained on MIMIC-III notes themselves. While this was arguably advantageous for both traditional fine-tuning and prompt learning, it may have overly favored prompt learning due to the

TABLE IX
BALANCED ACCURACY FOR PROMPT LEARNING AND TRADITIONAL FINE-TUNING WITH NONCLINICAL PLMS ON THE ICD-9 TRIAGE TASK. THESE RESULTS USED A SAMPLE SIZE OF 128 PER CLASS FOR TRAINING AND EVALUATED ON THE WHOLE TEST SET

Model	Traditional fine-tuning	Prompt learning
bert-base-uncased	0.2541	0.6301
roberta-base	0.3451	0.7989
gpt2	0.3812	0.8613
opt-125m	0.3233	0.7231

reformulation of the classification task as an MLM objective. Therefore, we present the results of another biomedical BERT model from Microsoft, the PubMedBERT, which was pretrained from scratch using abstracts from PubMed [45] in Table VIII. To a similar end, we also present results for nonclinical PLMs of a similar model size in Table IX. It can be seen that prompt learning still outperformed traditional fine-tuning by a large margin on the ICD-9 Triage task, in line with our other results.

ACKNOWLEDGMENT

REFERENCES

[1] L. Gao et al., "The pile: An 800GB dataset of diverse text for language modeling," 2021, *arXiv:2101.00027*.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.* Minneapolis, MI, USA: Association for Computational Linguistics, vol. 1, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[3] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, p. 140, Jan. 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html

[4] T. B. Brown et al., "Language models are few-shot learners," 2020, *arXiv:2005.14165*.

[5] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does BERT look at? An analysis of BERT's attention," in *Proc. ACL Workshop BlackboxNLP, Analyzing Interpreting Neural Netw. (NLP)*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 276–286. [Online]. Available: https://www.aclweb.org/anthology/W19-4828

[6] A. Chowdhery et al., "PaLM: Scaling language modeling with pathways," 2022, *arXiv:2204.02311*.

[7] OpenAI, "GPT-4 technical report," 2023, *arXiv:2303.08774*.

[8] T. Wolf et al., "Transformers: State-of-the-art natural language processing," in *Proc. Conf. Empirical Methods Natural Lang. Syst. Demonstrations*, 2020, pp. 38–45. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.6

[9] W. Han, B. Pang, and Y. N. Wu, "Robust transfer learning with pretrained language models through adapters," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 854–861. [Online]. Available: https://aclanthology.org/2021.acl-short.108

[10] O. Rohanian et al., "Lightweight transformers for clinical natural language processing," 2023, *arXiv:2302.04725*.

[11] J. Lee et al., "BioBERT: A pre-trained biomedical language representation model for biomedical text mining," 2019, *arXiv:1901.08746*.

[12] K. Huang, J. Altosaar, and R. Ranganath, "ClinicalBERT: Modeling clinical notes and predicting hospital readmission," 2019, *arXiv:1904.05342*.

[13] E. Lehman et al., "Do we still need clinical language models?" 2023, *arXiv:2302.08091*.

[14] E. Alsentzer et al., "Publicly available clinical BERT embeddings," in *Proc. 2nd Clin. Natural Lang. Process. Workshop*. Minneapolis, MI, USA: Association for Computational Linguistics, Jun. 2019, pp. 72–78. [Online]. Available: https://aclanthology.org/W19-1909

[15] Y. Peng, S. Yan, and Z. Lu, "Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets," in *Proc. 18th BioNLP Workshop Shared Task*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 58–65. [Online]. Available: https://aclanthology.org/W19-5006

[16] S. Gururangan et al., "Don't stop pretraining: Adapt language models to domains and tasks," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 8342–8360. [Online]. Available: https://aclanthology.org/2020.acl-main.740

[17] M. Senior et al., "Identifying predictors of suicide in severe mental illness: A feasibility study of a clinical prediction rule (oxford mental illness and suicide tool or OxMIS)," *Frontiers Psychiatry*, vol. 11, p. 268, Apr. 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/fpsyt.2020.00268

[18] N. Vaci et al., "Real-world effectiveness, its predictors and onset of action of cholinesterase inhibitors and memantine in dementia: Retrospective health record study," *Brit. J. Psychiatry*, vol. 218, no. 5, pp. 261–267, May 2021.

[19] S. Chen, Y. Hou, Y. Cui, W. Che, T. Liu, and X. Yu, "Recall and learn: Fine-tuning deep pretrained language models with less forgetting," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 7870–7881. [Online]. Available: https://aclanthology.org/2020.emnlp-main.634

[20] S. Gururangan, S. Swayamdipta, O. Levy, R. Schwartz, S. Bowman, and N. A. Smith, "Annotation artifacts in natural language inference data," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.* New Orleans, LA, USA: Association for Computational Linguistics, Jun. 2018, pp. 107–112. [Online]. Available: https://aclanthology.org/N18-2017

[21] T. Niven and H.-Y. Kao, "Probing neural network comprehension of natural language arguments," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 4658–4664. [Online]. Available: https://aclanthology.org/P19-1459

[22] M. Hofer, A. Kormilitzin, P. Goldberg, and A. Nevado-Holgado, "Few-shot learning for named entity recognition in medical text," 2018, *arXiv:1811.05468*.

[23] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 4582–4597. [Online]. Available: https://aclanthology.org/2021.acl-long.353

[24] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," 2021, *arXiv:2107.13586*.

[25] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, pp. 3045–3059. [Online]. Available: https://aclanthology.org/2021.emnlp-main.243

[26] B. Paranjape, J. Michael, M. Ghazvininejad, L. Zettlemoyer, and H. Hajishirzi, "Prompting contrastive explanations for commonsense reasoning tasks," 2021, *arXiv:2106.06823*.

[27] X. Liu et al., "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," 2021, *arXiv:2110.07602*.

[28] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," 2022, *arXiv:2201.11903*.

[29] N. Shinn, F. Cassano, B. Labash, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: An autonomous agent with dynamic memory and self-reflection," 2023, *arXiv:2303.11366*.

[30] V. Sanh et al., "Multitask prompted training enables zero-shot task generalization," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–216. [Online]. Available: https://openreview.net/forum?id=9Vrb9D0WI4

[31] M. Moradi, K. Blagec, F. Haberl, and M. Samwald, "GPT-3 models are poor few-shot learners in the biomedical domain," 2021, *arXiv:2109.02555*.

[32] S. Sivarajkumar and Y. Wang, "HealthPrompt: A zero-shot learning paradigm for clinical natural language processing," 2022, *arXiv:2203.05061*.

[33] N. Houlsby et al., "Parameter-efficient transfer learning for NLP," 2019, *arXiv:1902.00751*.

[34] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," 2021, *arXiv:2106.09685*.

[35] T. Schick and H. Schütze, "Exploiting cloze-questions for few-shot text classification and natural language inference," in *Proc. 16th Conf. Eur. Chapter Assoc. Comput. Linguistics*, 2021, pp. 255–269. [Online]. Available: https://aclanthology.org/2021.eacl-main.20

[36] A. E. W. Johnson et al., "MIMIC-III, a freely accessible critical care database," *Sci. Data*, vol. 3, no. 1, p. 5, May 2016.

[37] B. van Aken, J.-M. Papaioannou, M. Mayrdorfer, K. Budde, F. A. Gers, and A. Löser, "Clinical outcome prediction from admission notes using self-supervised knowledge integration," 2021, *arXiv:2102.04110*.

[38] Z. Yuan, C. Tan, and S. Huang, "Code synonyms do matter: Multiple synonyms matching network for automatic ICD coding," 2022, *arXiv:2203.01515*.

[39] S. Wang, M. B. A. McDermott, G. Chauhan, M. Ghassemi, M. C. Hughes, and T. Naumann, "MIMIC-extract: A data extraction, preprocessing, and representation pipeline for MIMIC-III," in *Proc. ACM Conf. Health, Inference, Learn.* New York, NY, USA: Association for Computing Machinery, Apr. 2020, pp. 222–235, doi: 10.1145/3368555.3384469.

[40] W. Boag, D. Doss, T. Naumann, and P. Szolovits, "What's in a note? Unpacking predictive value in clinical note representations," *AMIA Summits Transl. Sci. Proc.*, vol. 2018, pp. 26–34, May 2018.

[41] R. Ma et al., "Template-free prompt tuning for few-shot NER," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.* Seattle, WA, USA: Association for Computational Linguistics, 2022, pp. 5721–5732. [Online]. Available: https://aclanthology.org/2022.naacl-main.420

[42] N. Ding et al., "OpenPrompt: An open-source framework for prompt-learning," 2021, *arXiv:2111.01998*.

[43] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost," in *Proc. 35th Int. Conf. Mach. Learn.*, J. Dy and A. Krause, Eds. vol. 80, Jul. 2018, pp. 4596–4604. [Online]. Available: https://proceedings.mlr.press/v80/shazeer18a.html

[44] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019, pp. 1–18. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7

[45] Y. Gu et al., "Domain-specific language model pretraining for biomedical natural language processing," *ACM Trans. Comput. Healthcare*, vol. 3, no. 1, pp. 1–23, Jan. 2022.

Niall Taylor received the bachelor's in experimental psychology from the University of Plymouth, Plymouth, U.K., in 2014, and the master's degree in experimental psychology from the University of Bristol, Bristol, U.K., in 2018. He is currently pursuing the D.Phil. degree in health data science with the University of Oxford, Oxford, U.K.

His research interests lie on the intersection between machine learning and health research, in particular natural language processing using large language models (LLMs) applied to clinical text data within electronic health records with a focus on computationally efficient and transparent methods.

Yi Zhang received the joint B.S. degree in mathematics and applied mathemathics from The University of Manchester, Manchester, U.K., and Tianjin University, Tianjin, China, in 2020, and the M.S. degree in mathematical sciences from the University of Oxford, Oxford, U.K., in 2022.

Since 2021, he has been a Research Scientist at the Department of Psychiatry, University of Oxford, where he has been working on several interdisciplinary projects from cellular imaging to electronic health records. His research interests include deep learning, natural language processing, and their applications.

Dan W. Joyce received the B.Sc. and Ph.D. degrees in computer science from the University of Southampton, Southampton, U.K., 1995 and 2001, respectively, and artificial intelligence and the MBBS (Medicine) degree in psychiatry from King's College London, London, U.K., in 2010.

He is currently an Honorary Consultant Psychiatrist at the Mersey Care NHS Foundation Trust, Prescot, U.K., and a Professor of connected mental health at the University of Liverpool, Liverpool, U.K. His research focuses on the application of computational methods to problems in clinical psychiatry.

Dr. Joyce is a member of the Royal College of Psychiatrists.

Ziming Gao received the B.Sc. degree in mathematics and applied mathematics from the University of Liverpool, Liverpool, U.K., in 2021, and the M.S. degree in mathematical sciences and mathematical finance (second master) from the University of Oxford, Oxford, U.K., in 2022.

His research focuses on the application of natural language processing methods in clinical psychiatry. His research interests include deep learning, reinforcement learning, natural language processing, and their applications.

Andrey Kormilitzin received the Ph.D. degree in theoretical physics from Tel Aviv University, Tel Aviv, Israel, in 2012.

He has been working on mathematical aspects of fundamental interactions and statistical machine learning. He is the author of the Med7—open-source python library for clinical natural language processing. His research interests include machine learning, neural networks, stochastic analysis, and bioinformatics.

Alejo Nevado-Holgado received the combined B.Sc. and M.Sc. degrees in computer science from the Complutense University of Madrid, Madrid, Spain, in 2007, the Ph.D. degree in computer science from the University of Bristol, Bristol, U.K., in 2010, and the combined B.Sc. and master's degree in physical sciences from National University for Distance Education (UNED), Madrid, in 2015.

He is currently the Lead of AI and bioinformatics at the Department of Psychiatry, University of Oxford, Oxford, U.K. He also holds a position at the Big Data Institute, University of Oxford. He and his teams have a strong track record on the applications of AI to the study of mental health and dementia.