BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Multi-task Learning with AdapterFusion

Master Thesis of

Anjali Grover
MATRICULATION NUMBER: 901129

Beuth University of Applied Sciences, Berlin
Department VI - Data Science
Data Science and Text-based Information Systems (DATEXIS)

First Reviewer:	Prof. Dr.-Ing. habil. Alexander Löser
Second Reviewer:	Prof. Dr. rer. nat. Felix Bießmann
Advisor:	Betty van Aken

August 26, 2021

Abstract

This work aims to make progress in developing Clinical Decision Support Systems by improving patient outcome predictions. We work towards further improving the recent progress made by Aken et al., 2021 (CORe) on clinical outcome predictions. Clinical outcome tasks use the patient's admission notes to predict outcomes like Diagnosis at discharge, procedures performed, in-hospital mortality, and patient's length of stay at the hospital. The knowledge of these outcomes can help medical professionals not overlook risks and plan hospital capacities. We compare and evaluate traditional multi-task learning methods against the novel approach of AdapterFusion on clinical outcome tasks to solve the challenges faced by the CORe approach. We present that the traditional multi-task method best predicts the procedures (% AUROC: 90.50). However, we observe worse performance on other tasks using Multi-task learning as it suffers from catastrophic interference problem. We further show that AdapterFusion suffers from an overfitting problem on extreme multi-label tasks: Diagnosis and Procedures. A detailed analysis reveals a high class variability in the dataset as a plausible reason for the poor performance of AdapterFusion models. On the other hand, we find that Single-task Adapters surpass baselines on Diagnosis (% AUROC: 83.98) and Length of Stay task (% AUROC: 75.45) and perform on par with the CORe approach on Procedures and Mortality Prediction. Additionally, Single-task adapters help solve problems like high training time and high resource usage faced by the traditional fine-tuned transformer-based models.

Contents

Chapter 1

Introduction

"Imagine if a doctor can get all the information she needs about a patient in 2 minutes and spend the next 13 minutes of a 15-minute office visit talking with the patient, instead of spending 13 minutes looking for information and 2 minutes talking with the patient."

In his book Deep Medicine, Eric Topol quotes Lynda Chin, a world-renowned cancer genomics scientist (Topol, 2019). Eric Topol is an American cardiologist and provides various examples to explain how shallow medicine practiced these days is detrimental to patients' health. He refers to shallow medicine as scenarios where the medical professionals spend little time with patients to understand their actual problems.

As per research (Irving et al., 2017), primary care doctor visits last less than five minutes for half of the world's population. It has further been associated with poorer health for patients and burnout for doctors. Moreover, as per another study (Singh, Meyer, and Thomas, 2014), 12 million people are misdiagnosed every year in the U.S.A. alone. The misdiagnosis leads to the death of 40,000 - 80,000 people annually. However, as Eric Topol explains in Deep Medicine, this problem can be solved using AI-based clinical decision support (CDS) systems. CDS uses Electronic Health Records (EHRs) of the patients and supports the doctors in pinpointing certain risk factors. It helps the medical professionals not overlook potential risks and focus on the patient's treatment instead of spending the entire time of an appointment adding data into the systems.

Therefore, in this work, we focus on predicting clinical outcome predictions using Electronic Health Records, which we hope can help the medical community in the future.

1.1 Objective

This work is based upon the novel task setup proposed by Aken et al., 2021 for clinical outcome prediction that simulates the patient's admission state and predicts the outcome of the current admission. The four clinical outcome tasks we focus on in this work include:

1. **Diagnosis prediction** In this task, we predict the patient's diagnosis using the admission notes. It is a primary task of clinical outcome predictions to support medical professionals with differential diagnoses.
2. **Procedure prediction** With this task, we predict different procedures that can be used to treat the patient. The knowledge of Diagnosis and Procedures at the time of patient admission can help doctors not overlook risks.
3. **In-hospital mortality prediction** In this work, we also predict the patient's mortality during the admission time. This knowledge can help the medical professionals in ICUs with making optimal clinical decisions
4. **Length-of-stay prediction** Lastly, in this work, we predict the length of a patient's stay in the hospital. It can help the medical staff in planning the hospital resources.

In Aken et al., 2021, the authors proposed the Clinical Outcome Representations (CORe) to integrate patient trajectories into the pre-trained transformer-based models.

They trained four separate models for each task and surpassed various existing baselines. However, the CORe approach posed high training time, high resource usage, and plausible overfitting problems. Additionally, as the tasks are trained separately, it is hard to understand if one task can help the second task improve its performance. Therefore, we experiment with Traditional Multi-task learning and AdapterFusion to overcome the challenges posed by the CORe approach.

1.2 Motivation

Multi-task Learning In Multi-task learning, we train one single model that predicts multiple tasks in parallel. It has been adapted from how human beings learn in their day-to-day lives. We discuss the different methods of Multi-task learning in detail in Chapter 2.

Advantages of Multi-task Learning In our work, which aims to improve the performance on clinical outcome prediction tasks using the CORe model, we present the following points that motivate us to incorporate Multi-task Learning in our work:

- **Reduced Risk of Overfitting**
  To solve clinical outcome tasks, the authors of the CORe model create separate models for each of the four tasks. Thus, the models tend to learn on training data too well and reduce performance on the unseen data. However, with the

multi-task learning approach, as the model aims to create representations that generalize each task, the risk of overfitting is reduced.

- **Shared Representations**
  With the approach of creating different models for all of the downstream tasks, we completely ignore the possibility of the four tasks learning from each other. Instead, through Multi-task learning, we aim to understand if a particular prediction for Task 1 can help it predict the outcome for Task 2.
  For example, *Can a Diagnosis of Heart Failure help the model understand the increase in the patient's mortality rate?*

- **Low Training Time and Resource usage**
  Creating different models for each task also leads to high training time and resources. In this work, we tackle this problem using Multi-task learning methods. Only one model is trained in a multi-task setup, so the model size and training time is low. Additionally, it helps in reducing the inference time, which is advantageous for hospitals with limited resources.

Challenges Indeed, Multi-task learning helps us resolve some of the critical challenges we face using the one model for one task approach. However, it also leads to some of the problems which we discuss below:

- **Catastrophic Interference**
  Catastrophic interference, also known as Catastrophic forgetting, is a process where the deep neural networks forget the old parameters on learning new parameters (McCloskey and Cohen, 1989). As a Multi-task learning setup involves the model learning new parameters from each task, the model can start forgetting its old parameters as more and more tasks are included in the setup. Hence, it is difficult for the model to perform equally well on each task with multi-task learning.

- **Retraining for new tasks**
  It is challenging to introduce new tasks in a multi-task setup, as complete retraining of the model is required. It involves finding the shared representations of the tasks again using techniques such as hyperparameter optimization.

AdapterFusion To overcome the challenges presented by the traditional multi-task learning methods, we explore a novel way of training tasks using AdapterFusion (Pfeiffer et al., 2021). We use Single-task Adapters with Fusion for our experiments and have been discussed in detail in Chapter 2. In brief, AdapterFusion employs compact and extensible Adapter modules. Adapter modules (Houlsby et al., 2019) use very few trainable parameters compared to the fully fine-tuned models, which leads to less training time without compromising the model's performance. AdapterFusion is a technique that helps find relevant task adapters for the target task. For example, AdapterFusion may help us find out if the adapter for the Procedures task is relevant in improving the performance of the Diagnosis task.

Motivation to use AdapterFusion AdapterFusion is the central part of our work in predicting clinical outcome tasks; we have dedicated this section to demonstrate our motivation for using AdapterFusion and comparing it with the traditional multi-task learning approach. We have listed below some of the critical points that inspired us to explore AdapterFusion in the clinical domain:

- **Low usage of Resources**
  With AdapterFusion, we have the flexibility to save only the weights of the fusion parameters after training the model, which dramatically decreases the storage space required to save the model. It directly solves the problem we have when training a separate model for each downstream task.
- **Mitigation of Catastrophic Interference**
  To evaluate if AdapterFusion resolves the problem of catastrophic interference posed by multi-task learning, the authors compared the performance of Single Task Adapters and Multi-task Adapters with the Fusion process (Figure 2.8). They presented that AdapterFusion improves the performance on the downstream task with both ST-A and MT-A. Therefore, AdapterFusion solves the problem of catastrophic forgetting posed by traditional Multi-task learning methods.
- **No Retraining is required for new tasks**
  As discussed in the previous section, Multi-task training requires simultaneous access to all the tasks. Thus, complete retraining is required when adding new tasks to the setup. The Single-Task AdapterFusion proposed by the authors solves this problem, as no complete joint retraining is required on adding new tasks.

To conclude, we aim to improve upon the progress made by the CORe approach in predicting clinical outcome tasks, namely, Diagnosis, Procedure, Mortality, and Length of Stay of the patient. We compare the Traditional Multi-task learning method with the novel AdapterFusion approach and evaluate them on challenges faced by the CORe approach.

1.3 Outline

This thesis is structured into six chapters. Chapter 2 lays the theoretical foundations. It presents the taxonomy of Transfer Learning in NLP and discusses the performance and architecture of the baseline models. We also discuss in detail different Multi-task learning approaches and architectures for Adapter modules and AdapterFusion. Chapter 3 introduces our methodology, which overviews the downstream tasks, datasets, and evaluation metrics. Chapter 4 presents our implementations in detail, while Chapter 5 shows our experiments, evaluates and discusses them. In the end, Chapter 6 concludes and gives an outlook of possible future work.

Chapter 2

Background and Related Work

This chapter explicates Transfer learning and pre-trained language models such as BERT, BioBERT, and CORe, which are the underlying paradigm of this work. We further illustrate traditional Multi-task learning methods and compare them with the novel approach of AdapterFusion. As our work is based upon evaluating the Multi-task Learning approach and AdapterFusion on clinical tasks, we also refer to their architecture details and the results presented by the respective authors.

2.1 Transfer Learning

In his thesis Ruder, 2019 aptly describes Transfer Learning by comparing it with classic supervised learning. He explains that if we desire to solve a task from domain D in supervised learning tasks, we are provided with the labeled data for the same task. However, the supervised learning paradigm breaks down when we do not have sufficient labeled data. In these scenarios, Transfer learning allows the usage of a related task or domain and transfers the knowledge to the target task. Furthermore, Ruder, 2019 introduces us to the taxonomy for Transfer Learning for NLP as depicted in Figure 2.1.

Our work focuses on Inductive Transfer learning and performs various experiments to compare Sequential Learning with Multi-task learning methods. We would first introduce sequential transfer learning where a pre-trained model M on task A is further fine-tuned to make predictions on task B. Next, we would discuss Multi-task learning, where, unlike sequential transfer learning, tasks are trained simultaneously to gain knowledge from each other and make better predictions. The following sections would further elucidate the workings of transfer learning and some of the pre-trained models used in our work.

2.1.1 BERT (Bidirectional Encoder Representations from Transformers)

Google AI introduced BERT (Devlin et al., 2019) to produce state-of-the-art results on a wide variety of Natural Language Processing tasks, including Question Answering

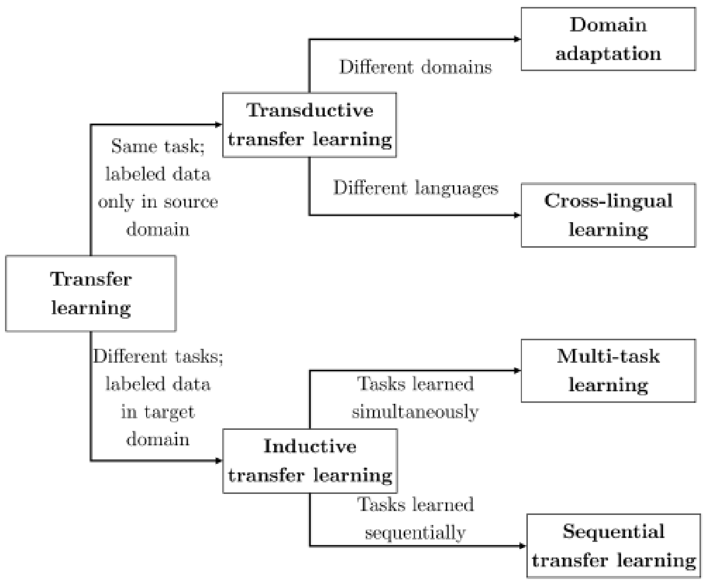

Transfer learning
Same task; labeled data only in source domain
Transductive transfer learning
Different domains
Domain adaptation
Different languages
Cross-lingual learning
Different tasks; labeled data in target domain
Inductive transfer learning
Tasks learned simultaneously
Multi-task learning
Tasks learned sequentially
Sequential transfer learning

FIGURE 2.1: A taxonomy for transfer learning for NLP Ruder, 2019, p. 64

(SQuAD v1.1), Natural Language Inference (MNLI), and others. Moreover, Google has also been using BERT to understand user searches better since 2019 Nayak, 2019.

BERT utilizes the concept of contextual relationships between word tokens from models such as ELMo (Peters et al., 2018) and ULMFit (Howard and Ruder, 2018). However, unlike other models, BERT is deeply bidirectional,i.e., it learns information from both the left and right sides of a token's context during the training. Additionally, BERT is pre-trained using large text corpora from BooksCorpus (800M words) and English Wikipedia (2,500M words).

BERT Architecture The Transformers model includes an encoder that takes a sentence as an input and a decoder that predicts the target task in its vanilla form. BERT, on the other hand, is a multi-layer bidirectional Transformer encoder. In their paper, the authors of BERT introduced the NLP community to two model sizes BERT$_{\text{BASE}}$ and BERT$_{\text{LARGE}}$. Table 2.1 represents the difference in the architectures of both models.

Model	*L*	*H*	*A*	*Total Parameters*
BERT$_{\text{BASE}}$	12	768	12	110 M
BERT$_{\text{LARGE}}$	24	1024	16	340 M

TABLE 2.1: Architecture details for BERT$_{\text{BASE}}$ and BERT$_{\text{LARGE}}$ models. **L** represents the number of layers, **H** means the hidden size and **A** represents the number of attention heads (Devlin et al., 2019).

BERT Input/Output Representation BERT's input representation is constructed using the sum of the token, segment, and position embeddings. A visualization of

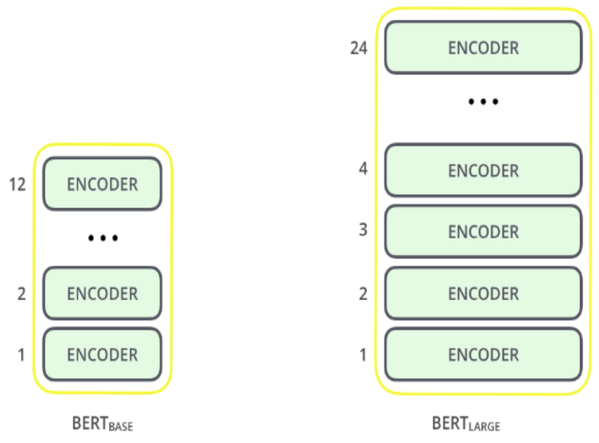


FIGURE 2.2: Encoder-based BERT Models. $\text{BERT}_{\text{BASE}}$(left) includes 12 layers of encoders, and $\text{BERT}_{\text{LARGE}}$ (right) contains 24 layers. (Alammar, 2018)

this construct is represented in Figure 2.3. As per the authors, an input sequence in BERT can be any contiguous text rather than a linguistic text. The sentences are separated from each other using the [SEP] token. The first token of each input sequence is [CLS], a Classification token representing the aggregated sequence representation to predict classification tasks. The authors of BERT essentially used two strategic unsupervised tasks to pre-train the model, which are discussed below:

Masked Language Model Standard Conditional models such as RNN and LSTM are trained either left-to-right or right-to-left because bi-directional conditioning would indirectly allow each word to see itself. Therefore, the authors of BERT used a novel idea of randomly masking 15% of the input words in a sequence and using the masked words' position to predict the input token.

Next Sentence Prediction In order to help BERT understand the relationship between different sentences, the authors used a binary next sentence prediction task to pre-train BERT. They chose sentences A and B for this task so that 50% of the time, sentence B followed A (labeled as IsNext), and the other 50% of the time, they used a random sentence from the corpus (labeled as NotNext). They demonstrated that pre-training BERT using this task was highly beneficial for Question Answering and Natural Language Inference tasks.

2.1.2 BioBERT (Bidirectional Encoder Representations from Transformers for Biomedical Text Mining)

BioBERT (Lee et al., 2019) is a variant of $\text{BERT}_{\text{BASE}}$, pre-trained on the biomedical domain corpora, unlike the original BERT model, which was trained using general domain texts such as Wikipedia. The authors of BioBERT used publicly available resources like PubMed Abstracts (4.5 B words) and PubMed Central Full-text articles

(13.5 B words) to incorporate biomedical knowledge into the model. BioBERT essentially helped the NLP community focused on biomedical tasks to get better word representations for bio-medical text. The authors demonstrated in their research that BioBERT achieved state-of-the-art performance on various biomedical text mining tasks. For example, BioBERT achieved higher F1 scores in biomedical NER (0.62) and biomedical RE (2.80) and a higher MRR score (12.24) in biomedical QA than the previous state-of-the-art models.

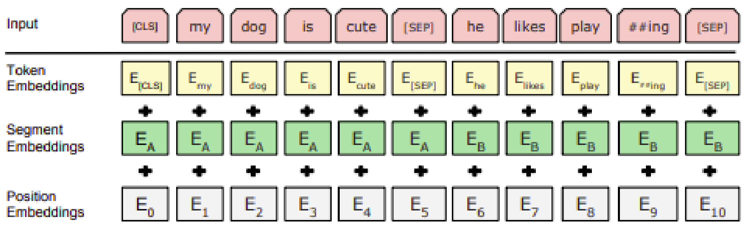

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

FIGURE 2.3: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings, and the position embeddings.Devlin et al., 2019

2.1.3 CORe (Clinical Outcome Representations)

Our work is primarily based upon the improvements made by the CORe model (Aken et al., 2021). The authors of CORe proposed Clinical Outcome Representations to integrate the patient representations into existing language models. We discuss their approach and its success in detail in the next section.

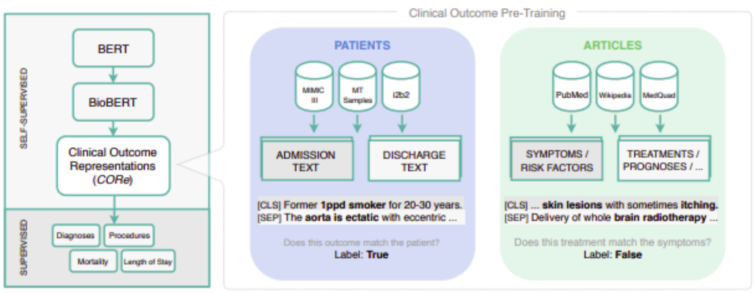

SELF-SUPERVISED
BERT
BioBERT
Clinical Outcome Representations (CORe)
SUPERVISED
Diagnoses
Procedures
Mortality
Length of Stay
Clinical Outcome Pre-Training
PATIENTS
MIMIC III
MT Samples
i2b2
ADMISSION TEXT
DISCHARGE TEXT
[CLS] Former 1ppd smoker for 20-30 years.
[SEP] The aorta is ectatic with eccentric ...
Does this outcome match the patient?
Label: True
ARTICLES
PubMed
Wikipedia
MedQuad
SYMPTOMS / RISK FACTORS
TREATMENTS / PROGNOSES / ...
[CLS] ... skin lesions with sometimes itching.
[SEP] Delivery of whole brain radiotherapy ...
Does this treatment match the symptoms?
Label: False

FIGURE 2.4: The CORe Approach (Aken et al., 2021)

CORe Approach Clinical Outcome Representations are created by pre-training on top of BioBERT weights. The idea behind the approach was to incorporate clinical knowledge to build a specialized model for outcome prediction. The authors used a new task setup related to the Next Sentence prediction task (Devlin et al., 2019) for the model to understand the relationship between patient admissions and outcomes. The task setup is specialized in inserting knowledge about clinical outcomes. As visualized in Figure 2.4, the patient's clinical notes were used to create Admission and Discharge sections. For 50% of the samples used for training, the correct Discharge section was followed by the Admission section (labeled as True). For the other 50%

samples, a negative sampling was applied (labeled as False). A similar procedure was used for medical articles where symptoms/risk factors replaced admissions, and discharge sections were replaced by treatments/prognosis.

Data Sources The authors of CORe used publicly available data to integrate patient and medical knowledge. They essentially divided their sources into two groups: *Patients and Articles*. It is inspired by the fact that doctors too learn from previous patients and medical literature.
CORe's Patients data source includes:

1. 32,721 discharge summaries from the MIMIC III training set
2. 5,000 publicly available medical transcriptions from the MTSamples website 2
3. 4,777 clinical notes from the i2b2 challenges 2006-2012

And, Articles are composed of:

1. 9,335 case reports from PubMed Central (PMC)
2. 2,632 articles from Wikipedia describing diseases
3. 1,467 article sections from the MedQuAd dataset extracted from NIH websites such as cancer.gov

Furthermore, the authors extracted admission and discharge sections from the above-mentioned data sources. For structured datasets such as MIMIC III, the rule-based approach was used to extract the required sections, whereas, for unstructured datasets like i2b2, a classifier proposed by Rosenthal, Barker, and Liang, 2019 was utilized.

CORe's Performance Table 2.2 represents the performance of CORe against baselines models on four clinical outcome prediction tasks. Our work primarily uses this model as the CORe model outperforms all existing baseline models for predicting clinical tasks. In addition, we employ novel approaches using multi-task learning to further improve upon the results of the CORe model in the following sections.

	Diagnosis (1266 classes)	*Procedures* (711 classes)	*In-Hospital Mortality* (2 classes)	*Length-of-Stay* (4 classes)
Bag of Words	75.87	77.47	79.15	65.83
Embeddings	75.16	76.72	79.94	66.78
CNN	61.18	73.13	75.50	64.49
BERT Base	82.08	85.84	81.13	70.40
Clinical BERT	81.09	86.15	82.20	71.14
Discharge BERT	82.86	87.09	84.51	71.73
BioBERT Base	82.81	86.36	82.55	71.59
CORe Articles	83.46	87.43	83.64	71.99
CORe Patients	83.41	**88.37**	83.60	71.96
CORe All	**83.54**	87.65	**84.04**	**72.53**

TABLE 2.2: Performance of CORe models against existing baselines in macro-averaged % AUROC. The CORe models outperform the baselines. This table is derived from Aken et al., 2021

2.2 Multi-Task Learning

The concept of Multi-task learning in Deep learning has been adapted from how human beings learn in general. On a day-to-day basis, we use the knowledge gained from the related tasks to learn new tasks. For example, from a programmer's perspective, we use concepts like functional programming from languages like C and C++ and apply the same understanding and skills when learning Java or Python. It works the same when learning a human language; for example, the knowledge one gains from learning Sanskrit can help the person learn Hindi faster and better.

Generally, In Machine Learning, we train a single model to perform the desired task. We use techniques such as Hyperparameter Optimization to find the best parameters for the model designed to solve a specific task. This method of focusing on a single task to train a model provides optimum results in most cases (Ruder, 2017); however, we lose information from the training signals of the related tasks. By sharing representation with related tasks, we can train our model to achieve higher performance on the target task (Ruder, 2017)

Hard-Parameter Sharing Hard-Parameter sharing is the most commonly used multi-task learning setup in Deep learning. The hidden layers between all the tasks are shared in this setup, whereas the task-specific layers are kept separate. Thus, it dramatically reduces the risk of overfitting; as more tasks are included in the setup, the model aims to find parameters representing all the tasks. Figure 2.5 presents a visual representation of the parameter sharing between different layers. Our work uses the Hard-parameter sharing method to evaluate traditional Multi-task Learning for predicting clinical tasks.

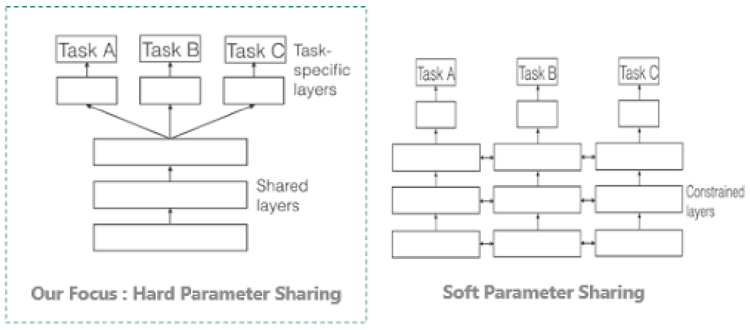


FIGURE 2.5: Commonly used Multi-task learning methods (Ruder, 2017)

Soft-Parameter Sharing Unlike Hard Parameter sharing, the Soft Parameter sharing method involves creating a separate model with separate parameters for each task. Most commonly, techniques like regularization are used to keep the distance between the parameters of different models low. As Hard-Parameter Sharing is the most commonly used method for Multi-Task Learning, we use just that for our experiments.

As discussed in Chapter 1, Multi-task learning helps us in reducing the risk of overfitting, understanding shared representations amongst different tasks, and leads to lower resource usage than the CORe approach. However, Multi-task learning also pose challenges like catastrophic interference.

To overcome the challenges presented by the traditional multi-task learning methods, we explore a novel way of training tasks using AdapterFusion (Pfeiffer et al., 2021), which is discussed in the following sections. However, before we discuss AdapterFusion, it is essential to understand the Adapter Modules and their architecture.

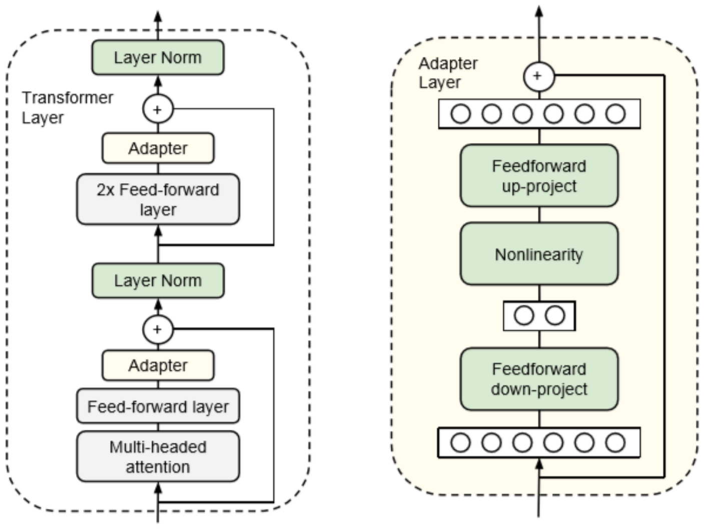


FIGURE 2.6: The left side represents the presence of the adapter modules twice in each transformer layer. The right side of the figure represents the adapter layer. The adapter consists of a bottleneck that contains few parameters relative to the attention and feedforward layers in the original model. Only the green layers are trained on the target task during adapter training, including the adapter, layer-normalization parameters, and the final classification layer. (Houlsby et al., 2019)

2.3 Adapter Modules

Houlsby et al., 2019 demonstrated Adapter Modules, a compact and extensible transfer learning method in NLP. The authors presented Adapter Modules as an alternative to scenarios where a new model is required for each downstream task. A standard fine-tuning process involves copying the parameters of the pre-trained model and tuning them to the downstream task. However, in cases where multiple downstream tasks are involved, the creation of each model is inefficient as it involves training millions of parameters per task. The idea behind the efficiency of the Adapters is that they require only a few parameters per task. The Adapter modules inject new parameters of dimension m to each transformer layer (Figure 2.6, left side) and keep the original parameters of the pre-trained model with dimension d intact, where m«d. During the training process with Adapter modules, only the adapter modules are changed.

As shown in Figure 2.6- right side, the adapters project the original feature size into a smaller dimension and then project it into the original size. Thus, it ensures that the number of parameters remains substantially small compared to the original pre-trained model.

In experiments with 26 diverse text classification tasks, the authors of Adapter Modules demonstrated that adapters yield stable performance even when substantially fewer parameters are used. For example, as shown in Figure 2.7, the performance of Adapter modules is better or similar to those of BERT models using only 3.6% of parameters per task, whereas fully fine-tuned models use 100% of the parameters.

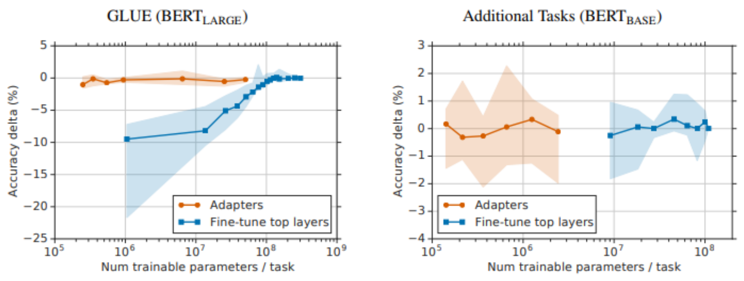


FIGURE 2.7: Accuracy versus the number of trained parameters aggregated across tasks. The authors compare adapters of different sizes (orange) with fine-tuning the top n layers for varying n (blue). The lines and shaded areas indicate the 20th, 50th, and 80th percentiles across tasks.(Houlsby et al., 2019)

2.4 Adapter Fusion

In 2021, AdapterFusion (Pfeiffer et al., 2021) was proposed to maximize knowledge transfer across tasks without suffering from the same problems as sequential fine-tuning and multi-task learning. Our work is inspired by the high performance of the AdapterFusion approach on a wide range of tasks, as shown in Figure 2.8. The authors of AdapterFusion experimented with two approaches for the Fusion Process: *Fusion with Single-Task Adapters (ST-A)* and *Fusion with Multi-task Adapters (MT-A)*. Fusion with ST-A refers to training adapters of different tasks separately and using them for the fusion process. Fusion with MT-A involves training adapters simultaneously with a multi-task objective and employs the fusion process as a next step. We further discuss the intrinsic details of the Adapter Fusion in the following sections.

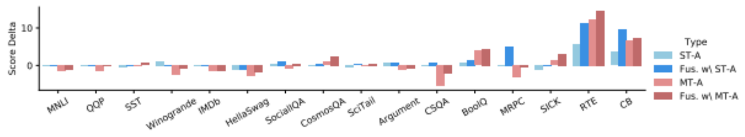


FIGURE 2.8: Relative performance difference of the two adapter architectures and the AdapterFusion models over fully fine-tuned BERT. Fusion improves over its corresponding adapters (ST-A and MT-A) for most tasks. (Pfeiffer et al., 2021)

Knowledge Extraction It is the first step in the AdapterFusion process where the adapters modules are trained separately for each task or in a multi-task setup. Our work focuses mainly on Single Task Adapters (ST-A), as it is demonstrated as the most efficient approach by the authors to share knowledge across all tasks. Therefore, for our work, we first train four adapter modules separately for each clinical outcome task.

In single task adapters, for each of the N tasks, the model is first initialized with parameters Θ_0. Additionally, as discussed in the previous section, randomly initialized adapter parameters Φ_n are introduced. During the training process, only adapter parameters, Φ_n, are used, and the original parameters, Θ_0, are kept intact. The objective for each task is to find the optimum weights for adapter parameters which would minimize the training loss. Mathematically, it can be represented in the following form:

$$\Phi_n \leftarrow \underset{\Phi}{argmin}[L_n(D_n; \Theta_0, \Phi)]$$

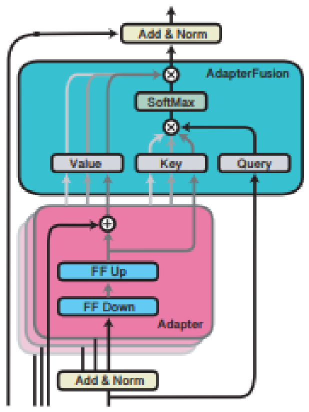
where D_n represents the data,
L_n represents the training loss,
Φ_n represents the adapter parameters
Θ_0 represents the original parameters of the pre-trained model

Knowledge Combination Knowledge Combination is the second step involved in the AdapterFusion process. In this novel step, the authors of AdapterFusion combine the adapters from the first step to overcome challenges such as catastrophic interference and training instabilities. They introduce Ψ that learns to combine N adapters to solve the target task. Mathematically, it is represented in the following form:

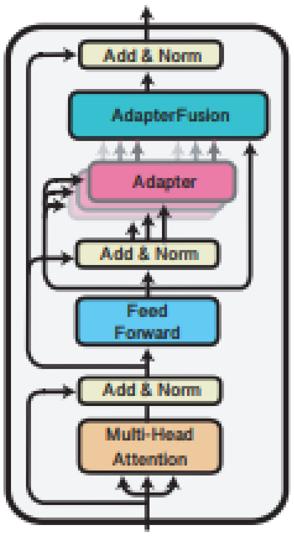
$$\Psi_m \leftarrow \underset{\Psi}{argmin}[L_m(D_m; \Theta, \Phi_1, ..., \Phi_n, \Psi)]$$

where Ψ_m are the AdapterFusion parameters of task m,
Θ represents the original parameters of the pre-trained model and refers to Θ_0 from the Knowledge Extraction step.

Inspired by attention modules (Vaswani et al., 2017) in the transformers model, the authors of AdapterFusion use the same concept to find the contextual activation of each adapter. As shown in Figure 2.9a, the AdapterFusion parameters consist of the Key, Query, and Value vectors. The query takes as input the output of the pre-trained transformer weights. Additionally, both Key and Value take as input the output of the respective adapters. Finally, the dot product of the query with all the keys is passed into a softmax function, which learns to weigh the adapters with respect to the context. Additionally, Figure 2.9b represents the AdapterFusion architecture inside a Transformer layer. The AdapterFusion component can be seen taking as input the representations of multiple adapters trained on different tasks and learning a parameterized mixer of the encoded information.



(A) AdapterFusion Architecture. (Pfeiffer et al., 2021)



(B) AdapterFusion architecture inside a Transformer. (Pfeiffer et al., 2021)

2.5 Summary

This chapter first presented the taxonomy of Transfer learning in NLP and emphasized the focus of this work, i.e., Sequential Transfer Learning and Multi-task learning. It then explicates the underlying paradigm models of this work, i.e., BERT, BioBERT, and the CORe approach. The chapter discussed the architecture and pre-training process of BERT and provided a brief introduction to BioBERT, a variant of BERT pre-trained on biomedical text. As our work is entirely based on the CORe approach's improvements on clinical outcome tasks, the chapter explained in detail its architecture and the pre-training process to integrate patient characteristics into the model. Furthermore, the chapter details the central part of this work, Multi-task Learning and AdapterFusion. The chapter first introduces the Multi-task learning process, its commonly used methods, advantages, and challenges. It focuses on the fact that Multi-task learning can help us overcome some of the challenges (Overfitting, High Training time, and Resource usage) posed when training a separate model for each clinical outcome task. Next, to overcome the challenges posed by Multi-task learning, we discuss the Fusion process and its performance on various tasks, which motivated us to apply it for our experiments on clinical tasks.

Chapter 3

Methodology

This chapter first focuses on detailing the problem description of our work. It discusses the data source used by us for the experiments, i.e., MIMIC III. It also sheds light on data pre-processing and the distribution of the class labels. Followed by the data pre-processing and distribution, this chapter details our approach to solve the problem, explaining how we use Multi-task learning and AdapterFusion for our experiments. Lastly, it describes the evaluation metrics used to measure our experiments.

3.1 Problem Definition

This work aims to improve the predictions made by the CORe approach (Aken et al., 2021) for clinical outcome predictions. In addition, our focus is to overcome problems such as overfitting, high training time, and resources. Moreover, the goal is to understand if a particular prediction for Task 1 can help it predict the outcome for Task 2. For example, *can a Diagnosis of Arthritis help the model understand the procedures required to operate it?*
We utilize clinical notes from MIMIC III for our experiments, a publicly available data source (Johnson, Pollard, and Shen, 2016, detailed overview in section 3.2.1). As represented in Figure 3.1 , we input the patients' clinical notes at admission, including their present illness, allergies, social & family history. Then, we use that information to train a model to predict four outcomes at discharge: Diagnosis, Procedures, In-Hospital Mortality, and Length of Stay of the patient.

3.2 Data Pre-Processing & Distribution

3.2.1 MIMIC III (Medical Information Mart for Intensive Care)

MIMIC III is a publicly available database (under a data use agreement) comprising information relating to patients admitted to critical care units at Beth Israel Deaconess Medical Center in Boston, Massachusetts (Johnson, Pollard, and Shen, 2016). This version was released in 2016 and is an update of the MIMIC II (Lee et al., 2011).

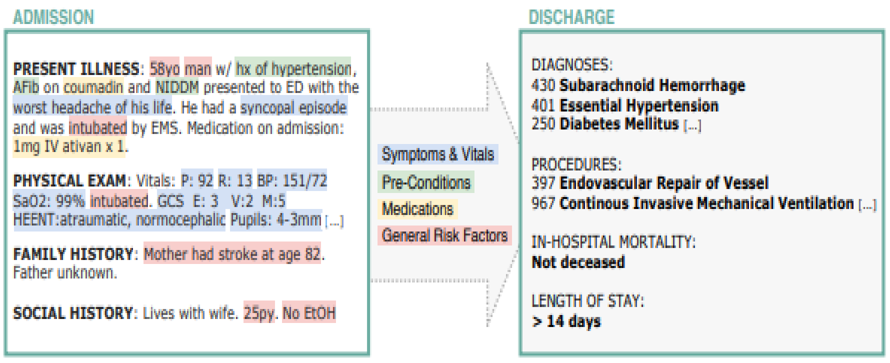

ADMISSION
PRESENT ILLNESS: 58yo man w/ hx of hypertension, AFib on coumadin and NIDDM presented to ED with the worst headache of his life. He had a syncopal episode and was intubated by EMS. Medication on admission: 1mg IV ativan x 1.
PHYSICAL EXAM: Vitals: P: 92 R: 13 BP: 151/72 SaO2: 99% intubated. GCS E: 3 V:2 M:5 HEENT:atraumatic, normocephalic Pupils: 4-3mm [...]
FAMILY HISTORY: Mother had stroke at age 82. Father unknown.
SOCIAL HISTORY: Lives with wife. 25py. No EtOH
Symptoms & Vitals
Pre-Conditions
Medications
General Risk Factors
DISCHARGE
DIAGNOSES:
430 Subarachnoid Hemorrhage
401 Essential Hypertension
250 Diabetes Mellitus [...]
PROCEDURES:
397 Endovascular Repair of Vessel
967 Continous Invasive Mechanical Ventilation [...]
IN-HOSPITAL MORTALITY:
Not deceased
LENGTH OF STAY:
> 14 days

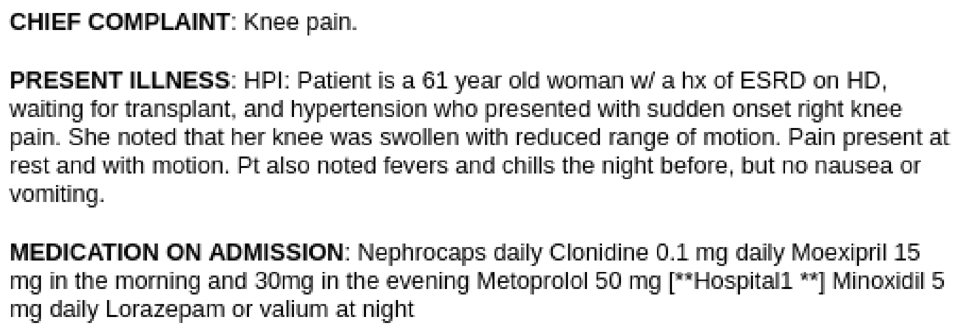
FIGURE 3.1: Admission to discharge sample demonstrating the outcome prediction task. (Aken et al., 2021)

MIMIC III data spans over a decade (2001 to 2012) and includes deidentified hospital admission information for 53,423 patients. It covers patient characteristics such as vital signs, medications, laboratory measurements, observations and notes charted by care providers, fluid balance, procedure codes, diagnostic codes, imaging reports, hospital length of Stay, survival data, and more. The MIMIC dataset was released to boost clinical research and education around the world. As a result, much research has been conducted using this data source. For example, Si and Roberts, 2019 use MIMIC III's clinical notes to propose a deep learning-based multi-task learning (MTL) architecture focusing on patient mortality predictions. Additionally, Sebastiano Barbieri, 2020 uses MIMIC III for predicting patient readmission to the ICU.

MIMIC III is provided as a collection of comma-separated value (CSV) files, in addition to the scripts to import the data directly into database systems like PostgreSQL and MySQL.

Admission Notes The Admission notes refer to the patient information a doctor has access to during the hospital admission time. It includes information such as *Chief Complaint*, *Present Illness*, *Medical History*, *Admission Medications*, *Allergies*, *Physical exam*, *Family history*, and *Social History* of the patients. We directly utilize the Admission notes created by Aken et al., 2021 from the MIMIC III data for the CORe approach. The authors of CORe used the following approach to extract Admission notes from MIMIC III:
The NOTESEVENTS.csv file from MIMIC III includes a clinical note (TEXT) for each Admission (HADM_ID) and Patient (SUBJECT_ID). The file also includes a CATEGORY column to extract only the "Discharge Summaries" for the Admissions. Further, using rule-based approaches, the authors of CORe extracted the relevant admission sections from the Discharge Summary Text. Figure 3.2 visualizes a sample Admission note used for our experiments. It has been derived from the public demonstration website[1] of Aken et al., 2021 and is not from the original MIMIC III data but of similar style and content.

[1] https://outcome-prediction.demo.datexis.com/



CHIEF COMPLAINT: Knee pain.

PRESENT ILLNESS: HPI: Patient is a 61 year old woman w/ a hx of ESRD on HD, waiting for transplant, and hypertension who presented with sudden onset right knee pain. She noted that her knee was swollen with reduced range of motion. Pain present at rest and with motion. Pt also noted fevers and chills the night before, but no nausea or vomiting.

MEDICATION ON ADMISSION: Nephrocaps daily Clonidine 0.1 mg daily Moexipril 15 mg in the morning and 30mg in the evening Metoprolol 50 mg [**Hospital1 **] Minoxidil 5 mg daily Lorazepam or valium at night

FIGURE 3.2: Sample Admission Note used for predicting clinical outcome tasks. The sample text is different from the actual text in MIMIC III data and has been derived from the public demonstration website of Aken et al., 2021.

Labels Like Admission notes, we use the class labels pre-processed by Aken et al., 2021 from MIIC III directly. The authors of CORe use the following approach to extract the labels for each clinical outcome task:
The diagnosis information corresponding to a patient's admission is retrieved using the *DIAGNOSES_ICD.CSV* file from MIMIC III. It includes ICD-9 diagnosis codes corresponding to each Admission and Patient. Similarly, ICD-9 codes for Procedures can be extracted using the *PROCEDURES_ICD.CSV* file from MIMIC III. ICD-9 codes (International Statistical Classification of Diseases and Related Health Problems,WHO, 1975) are the standardized numerical codes, first introduced by the World Health organization in 1975 to identify diseases, symptoms, medical procedures, and more. It is important to note that ICD-9 code length can vary from three to five digits. The length of ICD-9 codes identifies the granularity of the information they hold. Therefore, five-digit codes would include more specific information of diagnosis than a three- digit code. Moreover, there can be more than just one Diagnosis and Procedures ICD-9 code for an Admission ID.
For the Mortality label, "HOSPITAL_EXPIRE_FLAG" from file *ADMISSIONS.CSV* is used; a value of 1 presents that the patient died during the hospitalization, and 0 represents that the patient did not die. Length of Stay label is created by calculating the difference between the DISCHTIME (Discharge Time) and ADMITTIME (Admission Time).

After pre-processing the MIMIC data, a single CSV is retained, including the clinical note and the corresponding Diagnosis, procedures, mortality prediction, and length of stay labels (Figure 3.3).

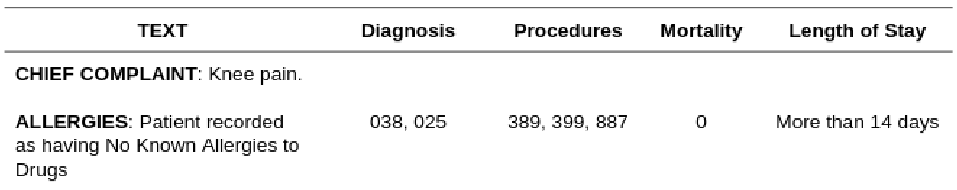
Data Distribution After pre-processing the MIMIC III data, CORe authors further employ a patient-wise split into train, validation, and test set with a 70/10/20 ratio. Figure 3.4 shows the statistics related to the Admission notes in MIMIC III data. Our work directly uses the dataset split created for CORe.

Diagnosis & Procedures As mentioned in section 3.2.1, ICD-9 codes range from three to five digits assigned to a unique category. Aken et al., 2021, groups the ICD-9 diagnosis codes from 4- into 3-digit codes to reduce complexity resulting in 1,266 diagnosis codes. Similarly, ICD-9 procedures codes were grouped into 3-digit codes. As represented in Figures 3.5a and 3.5b, the distribution of ICD-9 diagnosis codes and procedure codes follow power law and are extreme multi-label classification tasks. In addition, Figure 3.6 show the count of ICD-9 codes for Diagnosis and Procedures per dataset split.

Mortality & Length of Stay The Length of Stay label was split into four major categories: Under three days, three to seven days, one week to two weeks, and more than two weeks. As only 10% of the data in MIMIC III indicates that the patient died in the hospital (labeled as 1), mortality prediction is a highly imbalanced binary classification task. Figure 3.7 shows the distribution of Length of Stay and Mortality per dataset split.

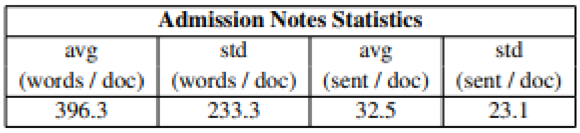
3.3 Our Approach

As discussed in Chapter 2, we perform experiments to evaluate the Traditional Multi-Task Learning approach compared to the AdapterFusion in order to predict clinical outcome tasks. To summarize, via Multi-task Learning, we aim to understand the shared representations between the four tasks, reduce the risk of overfitting and reduce the storage resources and training time. However, Multi-task learning can lead to catastrophic interference, worsening the performance on different tasks. On the other hand, AdapterFusion aims to mitigate catastrophic interference in addition to low training time and resource usage. We employ CORe as our underlying model for both setups, as it currently outperforms all existing baselines for predicting the novel clinical outcome tasks. We further discuss the methodology used to experiment with Multi-task Learning and AdapterFusion.



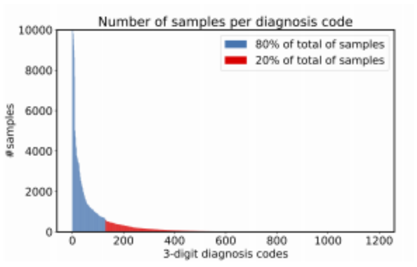
TEXT	Diagnosis	Procedures	Mortality	Length of Stay
CHIEF COMPLAINT: Knee pain. **ALLERGIES**: Patient recorded as having No Known Allergies to Drugs	038, 025	389, 399, 887	0	More than 14 days

FIGURE 3.3: Sample format of data after pre-processing. TEXT column includes the Admission notes for the patients. Diagnosis, Procedures, Mortality, and Length of Stay represent the labels for the four clinical outcome tasks.

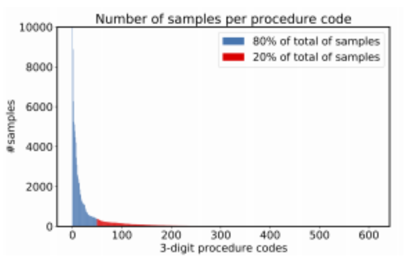


Admission Notes Statistics			
avg (words / doc)	std (words / doc)	avg (sent / doc)	std (sent / doc)
396.3	233.3	32.5	23.1

FIGURE 3.4: Numbers of words / sentences in MIMIC III admission notes. (Aken et al., 2021)



(A) Distribution of ICD-9 Diagnosis codes. (Aken et al., 2021)



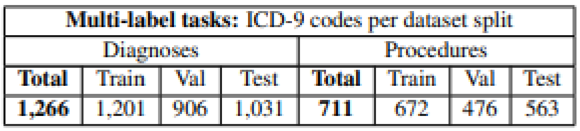
(B) Distribution of ICD-9 Procedures codes. (Aken et al., 2021)

Traditional Multi-task Learning

To better understand how Multi-task Learning affects the performance amongst the four tasks, we adopted the approach of including tasks in the experiments step by step. For example, we first start with a pair-wise experiment, including Diagnosis and Procedures. Then, in the next experiment, we add a third task to the experiment: Mortality Prediction. It helps us understand how related the tasks are to each other. If the performance of Diagnosis and Procedures peaks when in the setup together but decreases on introducing the Mortality prediction task, it clarifies if Mortality prediction task can help predict Diagnosis/Procedures task better.
In total, we perform eight different experiments using the traditional multi-task learning method including the following task settings:

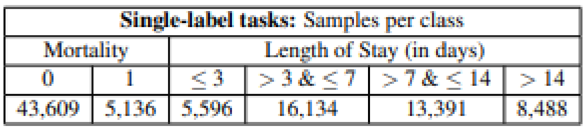
1. Diagnosis - Procedures
2. Diagnosis - Length of Stay
3. Diagnosis - Mortality
4. Procedures - Length of Stay
5. Procedures - Mortality
6. Diagnosis - Procedures - Mortality
7. Diagnosis - Procedures - Length of Stay
8. Diagnosis - Procedures - Mortality - Length of Stay

Please note that we do not perform experiments including just Mortality Prediction and Length of Stay in a setting, as admissions resulting in the patient's death would not have a label for the Length of Stay task. Therefore, a model would not learn additional parameters for better prediction in the "Length of Stay - Mortality Prediction" task setting.



Multi-label tasks: ICD-9 codes per dataset split							
Diagnoses				Procedures			
Total	Train	Val	Test	**Total**	Train	Val	Test
1,266	1,201	906	1,031	**711**	672	476	563

FIGURE 3.6: Distribution of ICD-9 codes per dataset split (patient-wise).



Single-label tasks: Samples per class					
Mortality		Length of Stay (in days)			
0	1	≤ 3	> 3 & ≤ 7	> 7 & ≤ 14	> 14
43,609	5,136	5,596	16,134	13,391	8,488

FIGURE 3.7: Distribution of Mortality and Length of Stay labels per dataset split (patient-wise).

Single Task Adapters (ST-A)

As discussed in Section 2.3, Adapters are lightweight, modular, and flexible. Before creating AdapterFusion models for each task, we create four Single-Task Adapters for each clinical outcome tasks. As the authors of Pfeiffer et al., 2021 recommended Fusion with Single-Task Adapters over Fusion with Multi-Task Adapters due to its high flexibility and high performance, we employ it for our experiments. In single task adapters, for each of the N tasks, the model is first initialized with parameters Θ_0. Additionally, randomly initialized adapter parameters Φ_n are introduced. During the training process, only adapter parameters, Φ_n, are used, and the original parameters, Θ_0, are kept intact. The objective for each task is to find the optimum weights for adapter parameters which would minimize the training loss. Mathematically, it can be represented in the following form:

$$\Phi_n \leftarrow \underset{\Phi}{argmin}[L_n(D_n; \Theta_0, \Phi)]$$

where D_n represents the data,
L_n represents the training loss,
Φ_n represents the adapter parameters
Θ_0 represents the original parameters of the pre-trained model

AdapterFusion

After creating ST-A for the four clinical tasks, we next utilize them for the AdapterFusion process. Inspired by attention modules (Vaswani et al., 2017) in the transformers model, the authors of AdapterFusion use the same concept to find the contextual activation of each adapter. They use Key, Query, and Value vectors to find the most relevant adapter of the related task at each layer. This in turn helps improve the performance on the target tasks.
For example: In AdapterFusion model for Diagnosis task, Diagnosis is the target task and the other tasks are the related tasks. The related tasks, i.e., Procedures,

Mortalitty, and Length of Stay aim to help increase the performance of the target task using the AdapterFusion process. More information on AdapterFusion's architecture is included in Section 2.4.

3.4 Evaluation Metrics

To measure the performance of our experiments against the CORe approach, we primarily use AUROC. However, before discussing AUROC, we introduce other evaluation metrics for classification tasks such as Precision, Recall, F1, and Confusion Matrix. We avail them in Chapter 5 for the Error Analysis of our experiments.

Confusion Matrix As represented in Figure 3.8, a confusion matrix is a table that helps us assess the performance of a model on the test data for a classification task. The four quadrants in the matrix represent True Negatives (TN), False Positives (FP), False Negatives(FN), and True Positives (TP).

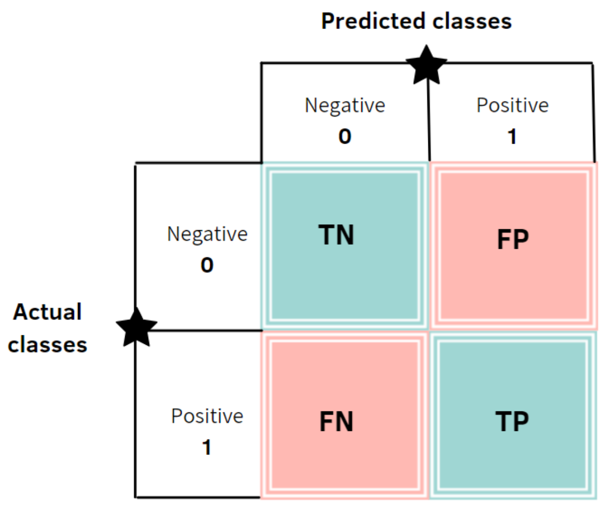


FIGURE 3.8: Confusion Matrix for a binary classification task. Rows present the actual labels, and columns present the models' predictions. The four quadrants represent true negative (TN), false positive (FP), true positive (TP), and false negative (FN) (Anello, 2021)

For example, if the task involves predicting the mortality of a patient, i.e., a binary classification task (label: 1/0). In this scenario, "True positives" represents the number of samples against which the model correctly predicted the patient would die during the hospitalization. "False positives" depicts the number of admission samples for which the model wrongly classified the patient would die during the hospitalization. Similarly, "True Negatives" represent the proportion of samples correctly classified as negative and "False Negative" depicts the number of positive examples that the model incorrectly classified as negative.

All of the above-mentioned terms are further used to describe metrics such as Precision, Recall, F1 score, and AUROC.

Precision It represents the accuracy of the positive predictions, i.e., the proportion of accurately classified samples out of all the records classified as positive.

$$Precision = \frac{TP}{TP + FP}$$

Recall It is also known as *True Positive rate*, and expresses the proportion of actual positives identified correctly:

$$Recall = \frac{TP}{TP + FN}$$

Generally, the evaluation metric is used according to the use case of the prediction task. In some cases, it is essential to reduce false positives, in which case we evaluate the precision metric. In other cases, it is crucial to identify all the positive cases correctly, hence, it is required to evaluate the Recall. However, it can be challenging to keep track of multiple evaluation metrics when it is vital to reduce both False Positives and False Negatives.

F1 score helps evaluate the models where the goal is to balance False Positives and False Negatives. F1 score is the harmonic mean of Precision and Recall and is represented using the following expression:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Macro and Micro Average We use Macro and Macro averages of the above-mentioned metrics in Chapter 5 for evaluating and comparing our different models. They are instrumental when dealing with multi-class labels, like in the case of our tasks.
In the calculation of Macro Average, equal weight is provided to each class and the results are averaged over each class in the data. However, For Micro average, a weighted average is calculated, and the weight depends on the distribution of the classes. More weight is provided to the class with a higher number of samples. Micro averages are of vital importance in the datasets with high class imbalance. Macro average and Micro average for a metric provide the same results for datasets including classes with equal distribution.

AUROC We essentially use AUROC to measure the performance of our experiments for a lateral comparison with the CORe approach. The AUROC is calculated as the area under the ROC curve. A ROC curve (Receiver Operating Characteristic Curve) shows the trade-off between True Positive Rate (TPR)/Recall and False Positive Rate (FPR) across different decision thresholds. It helps us understand our model's performance against a random classifier.

False Positive rate (FPR) is expressed as,

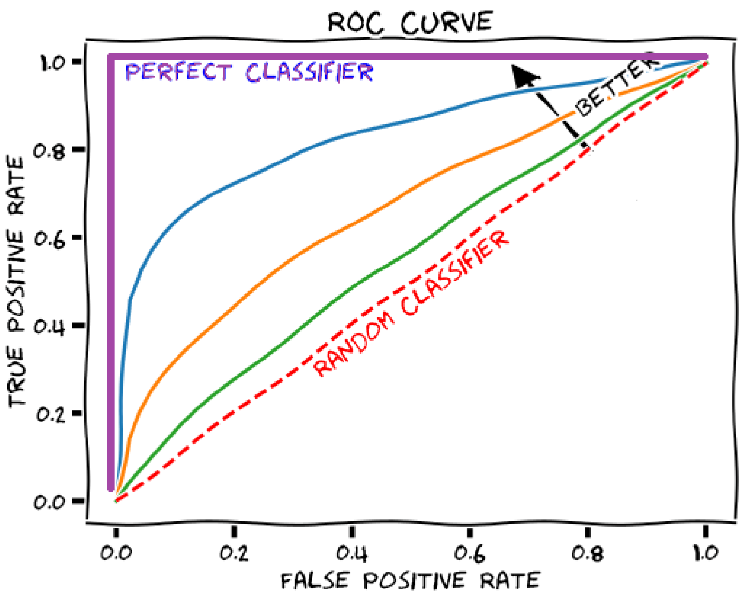


FIGURE 3.9: A sample ROC curve, Draelos, 2019

$$FPR = \frac{FP}{FP + TN}$$

For a random classifier, AUROC is 0.5 (Area under the Red line, Figure 3.9) and is the worst score. The best AUROC score is 1.0 (Area under the purple line, Figure 3.9), representing that the model can discriminate between the positive and negative sample correctly. An AUROC less than 0.70 is sub-optimal performance, 0.70 – 0.80 and greater than 0.80 are considered as good and excellent performance, respectively.

We would discuss the above-mentioned evaluation metrics at length to measure the performance of our experiments in Chapter 5

3.5 Summary

In this chapter, we first reiterated the goal of our work, i.e., to experiment with Multi-task Learning and AdapterFusion for better predictions on clinical outcome tasks. Next, it described our approach to solve this problem, i.e., we add tasks step by step in our experiments for Traditional Multi-task learning. Moreover, for the Adapter-Fusion process, we utilize *"Fusion with Single-task Adapters"* for its high flexibility and performance. Furthermore, we introduced our data source, MIMIC III, and elucidated the data pre-processing process. Additionally, we represented the data distribution of the labels and their split into the Train, Validation, and Test datasets. The data distribution helped interpret the imbalance of the classes for all four clinical outcome tasks. Lastly, we discussed evaluation metrics used by us to measure the performance of our experiments, namely, Precision, Recall, F1 score, Confusion Matrix, and AUROC.

Chapter 4

Implementation

In this chapter, we will focus on explaining the environment of our experiments, and the python libraries we used to build our models. Additionally, we will explain the Hyperparameter Optimization process for our experiments.

4.1 Experimental Environment

For our experiments, we use the DATEXIS Kubernetes cluster. It consists of 30 nodes with about 2,288 CPU cores, where some nodes are additionally equipped with GPUs. In total, there are 8 Nvidia Tesla K80, 8 Nvidia Tesla P100, 9 Nvidia Tesla V100, and 17 DGX A100 GPUs. Our experiments are executed with one Nvidia Tesla V100 GPU or using two Nvidia Tesla P100 GPUs.We use Docker containers based on Python version 3.7.5, CUDA13 version 10.1, and PyTorch version 1.6.0.

4.1.1 Implementing Transformers

Hugging Face

Hugging Face[1] is an open-source provider of Natural language processing technologies. The libraries we use for implementing Traditional Multi-task learning and AdapterFusion are based upon the Transformers library [2] from Hugging Face. Transformers library provides general-purpose architectures for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages. Additionally, the Hugging Face Hub allows researchers worldwide to share their models and datasets freely with each other.

In further sections, we describe the open-source libraries we use for Multi-task learning and the AdapterFusion process.

[1] https://huggingface.co/
[2] https://huggingface.co/transformers/

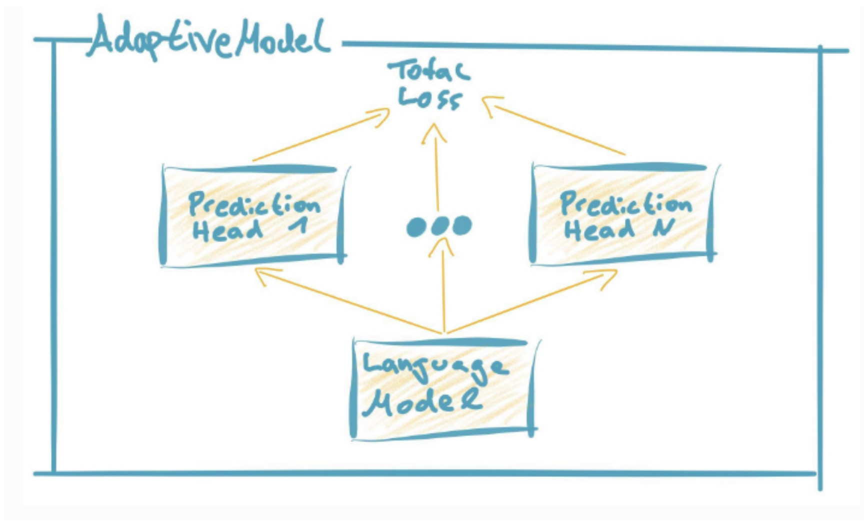


FIGURE 4.1: The Adaptive model architecture from FARM

4.1.2 Multi-task Learning

FARM

We use FARM v0.5.0 [3] to implement our traditional multi-task learning experiments using a hard-parameter sharing method. FARM is also an open-source library for NLP. FARM is built upon the Transformers library and provides additional features to simplify transfer learning implementation and debugging for developers. FARM provides full Compatibility with HuggingFace Transformers' models and model hub.

Adaptive Model with FARM

FARM was an optimal choice for us to implement Multi-task learning due to its Modular design of language models and prediction Heads. Figure 4.1 depicts FARM's flexibility of using one language model (for example, BERT) with multiple prediction heads to form an Adaptive Model.

- The Language model represents their class to include any pre-trained model such as BERT, BioBERT, CORE which convert word tokens from the text corpus into a vector representation.
- The Prediction head class converts the vector representations from the Language model into the predictions for the downstream task, for example, Mortality prediction.
- Adaptive Model allows using one Language Model with one or multiple prediction heads. For Multi-task Learning, we use multiple heads (one for each task in the experimental setup) with a language model (CORe).

[3] https://github.com/deepset-ai/FARM

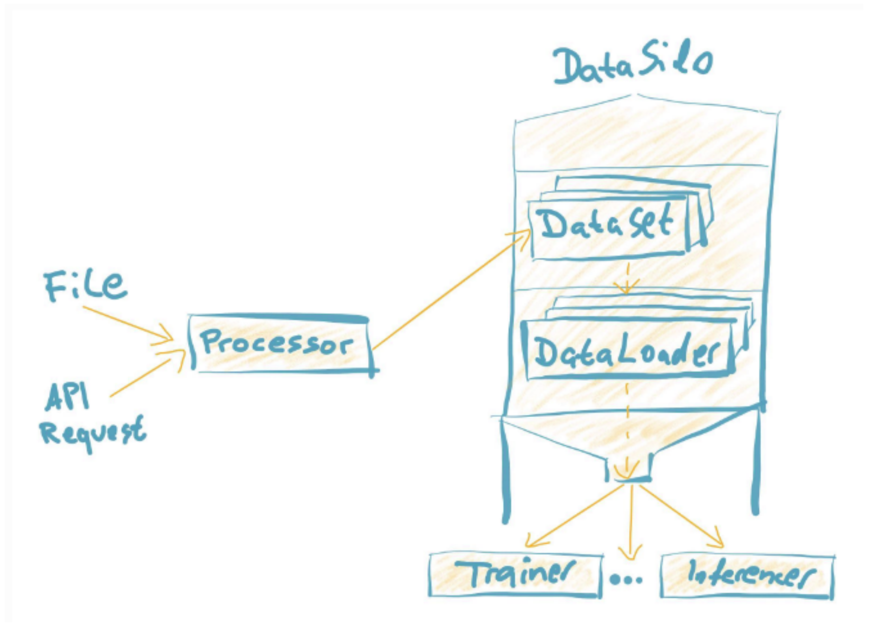


FIGURE 4.2: The Data Handling architecture from FARM

Data Processing with FARM

In addition to the Adaptive Model, FARM allows easy pre-processing of Custom datasets for NLP tasks. FARM has a generic pipeline at the backend, which allows the data to flow from one stage to another, leading to better debugging capabilities for the developers. Figure 4.2 showcases the Processor class of FARM.FARM is built upon the Transformers library and provides additional features to simplify transfer learning implementation and debugging for developers. Data Silo is a generic class used to load the train, validation, and test datasets and exposes a data loader for each dataset split.

4.1.3 Single-task Adapters and Adapter Fusion

Adapter-Transformers

We used the Adapter-Transformers v1.1 [4] library in this work for implementing Single-task Adapter modules and AdapterFusion. Like FARM, adapter-transformers are also built upon the Transformers library from HuggingFace. However, unlike FARM, adapter-transformers can be used as a drop-in replacement for HuggingFace Transformers and regularly synchronizes new upstream changes. Therefore, essentially, most of the files in the adapter-transformers are the same as that of the original Transformers library. The adapter-transformers extend the transformers library by integrating the Adapter modules into state-of-the-art language models. Similar to Transformer's Model Hub, AdapterHub [5] allows researchers to share pre-trained adapters freely.

[4] https://github.com/Adapter-Hub/adapter-transformers
[5] https://adapterhub.ml/

```python
from transformers import AutoModelWithHeads
model = AutoModelWithHeads.from_pretrained("bert-base-uncased")
model.load_adapter("sentiment/sst-2@ukp")
model.set_active_adapters("sst-2")
```

LISTING 4.1: Sample code for Inference using Adapter-Transformers library

```python
from transformers import AutoModelWithHeads

model = AutoModelWithHeads.from_pretrained("bert-base-uncased")
model.load_adapter("sts/qqp@ukp", with_head=False)
model.load_adapter("nli/qnli@ukp", with_head=False)

model.add_classification_head("cb")

adapter_setup = Fuse("qqp", "qnli")
model.add_fusion(adapter_setup)
model.set_active_adapters(adapter_setup)
model.train_fusion(adapter_setup)
```

LISTING 4.2: Sample code for Training AdapterFusion using Adapter-Transformers library

As adapter-transformers' files are very similar to the Transformers library, it is easy to understand the work flow and use the Adapter modules efficiently. Listing 4.1 shows that the format of working with transformers-based models like BERT is similar to the original Transformers library. The extra lines of codes involve loading the required adapter for the inference. $model.set\_active\_adapters("sst-2")$ allows the usage of just Adapter parameters during model's forward passes and freezes the parameters of the underlying language model, BERT.

It is a fairly simple process to implement the AdapterFusion process using the pre-trained Single-Task Adapters. As represented in Listing 4.2, we load the required pre-trained single task adapters and add a classification head for predictions on the target task. Next the *Fuse* composition block from the adapter-transformers library is used to combine the required adapters followed by add_fusion function to add the fusion layer to the setup. Finally, *set_active_adapters* allows only fusion layer's weights to be updated during the training process.

Our experiments with both, FARM and adapter-transformers library used Hyperparameter Optimization process to improve the performance on the target tasks.

4.2 Hyperparameter Optimization (HPO)

A model hyperparameter is a characteristic of a model whose value cannot be estimated from data. The value of the hyperparameter is required to be set before the machine learning process begins (Brownlee, 2017).

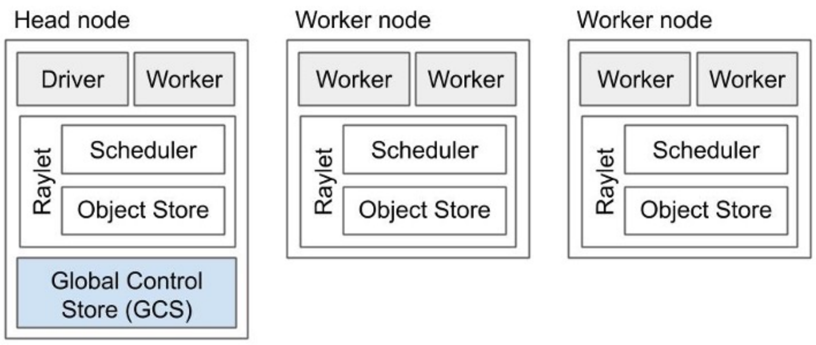


FIGURE 4.3: The architecture for Ray Cluster. Ray cluster allows the usage of multiple machines for an experiment using Head nodes and Worker nodes.

Hyperparameter optimization is a process of choosing optimal parameters for a machine learning algorithm. It is also called the fine-tuning process, which allows an algorithm to provide the best results on a specific task. In our experiments, we use HPO to find optimal values for the following parameters:

Traditional Multi-Task learning:

- learning_rate
- warmup_steps
- gradient_accumulation_steps
- embedding_dropout
- balance_classes

ST-A and AdapterFusion:

- learning_rate
- warmup_steps
- gradient_accumulation_steps

Before detailing the implementation of HPO for our experiments, we provide a brief introduction to the HPO parameters used for Traditional Multi-task learning and Adapter modules below:

- The learning_rate configures the step size of the Model while converging to solve a problem.
- The warmup_steps are a few steps the Model uses with a lower learning rate at the beginning of the training process.
- Gradient_accumulation_steps are the number of iterations the Model needs to wait before updating the weights of the variables. It simulates a larger batch size, even if it does not fit on the machine. The accumulated gradients over the mentioned steps are then used to configure the weights of the parameters.
- Embedding_dropout represents the dropout at the embedding layer to reduce overfitting.

- balance_classes (Boolean variable)checks if there is a need for class weights to counter the imbalanced data
- In addition to the hyperparameters mentioned above, we use task weights for Traditional Multi-task learning with FARM. They were utilized in order to tune the weight each task should contribute to each model parameter update.

Table 4.1 and 4.2 represent the search space used for Multi-task learning and Adapter-Fusion experiments, respectively.

	Search Space
learning Rate	[1e-4, 1e-6]
Warmup Steps	[50, 1500]
Gradient Accumulation Steps	[1, 20]
Embedding Dropout	[0.1, 0.3]
Balance Classes	[True, False]
Task weights	[0.001, 1]

TABLE 4.1: Search Space used for Hyperparameter Optimization in Traditional Multi-task learning (FARM) experiments.

	Search Space
learning Rate	[1e-4, 1e-6]
Warmup Steps	[50, 1500]
Gradient Accumulation Steps	[1, 20]

TABLE 4.2: Search Space used for Hyperparameter Optimization in Single-Task Adapters and AdapterFusion experiments.

4.2.1 ASHA

The most common approach to perform HPO is the Random-search and Grid-search method. Random search chooses the sets of hyperparameters randomly in a given search space. On the other hand, the Grid search technique builds a model for every combination of hyperparameters specified and evaluates each model. However, these techniques do not guarantee to find the optimal hyperparameters, and are compute-expensive. Moreover, the HPO process becomes more complex with the high dimensionality of the search space.

Therefore, we use ASHA (Asynchronous Successive Halving for the parallel setting, Li et al., 2018) scheduler to find hyperparameters for our models. ASHA is a State-of-the-art technique for Hyperparameter optimization. It exploits distributed computing and aggressively terminates the low-performing trials[6]. It helps save the computing resources and time for exploring the parameters that produce better results on the downstream tasks. The authors of ASHA presented results demonstrating that ASHA outperforms state-of-the-art methods such as, Fabolas, Population Based Training, BOHB, and Vizier in a suite of HPO benchmarks.

[6] A trial refers to a single set of parameters used to evaluate the model's performance from a given search space

4.2.2 Ray Tune

We employ Ray v1.4.1 to implement an ASHA scheduler for the Hyperparameter Optimization of our models. We used the Ray cluster[7] to compute multiple trials in parallel for one experiment. It allowed us to complete Hyperparameter Optimization experiments faster than what would have been the case with just one machine for all HPO trials. As shown in Figure 4.3, a ray cluster includes a head node and a set of worker nodes. The cluster's head node is started first, and then the worker nodes use the address of the head node to form a cluster. After a job is submitted on the head node, the worker nodes run different trials in parallel, resulting in faster execution of the experiment.

As mentioned in the beginning of this section, we use the DATEXIS Kubernetes cluster for our experiments. Our experiments use 3 Nvidia V100 GPUs in parallel for the Hyperparameter Optimization. 1 GPU is used to evaluate the model performance of a single trial.

4.3 Summary

This chapter focused on describing the experimental environment of our experiments. In the first section, we first described the open-source transformer-based libraries used for the experiments. We explained the FARM's data handling and model architecture in detail, which makes it an optimal environment for implementing Multi-task learning. Additionally, we discussed the adapter-transformers library we used to experiment with AdapterFusion. Lastly, we provided details into the Hyperparameter Optimization process using the ASHA algorithm for our experiments, followed by the Ray cluster setup used for distributed computing.

[7] https://docs.ray.io/en/latest/cluster/index.html

Chapter 5

Benchmark, Evaluation, and Discussion

In this chapter, we discuss the hypothesis stated by this work. We compare the performance of Baseline models against Multi-task Learning, Single-Task Adapters and AdapterFusion. Additionally, we perform a deep dive analysis to understand the performance of AdapterFusion models and also assess the quality of predictions for Diagnosis task by the different learning algorithms. Lastly, we summarize our insights from the experiments in the discussion section.

5.1 Hypothesis

This work aims to implement the Traditional Multi-task learning and AdapterFusion for predicting clinical outcome tasks. Additionally, this work endeavors to compare the performance of the two approaches on the target tasks. Additionally, we strive to understand the inter-contextual representations of the different tasks.
Based on the advantages and challenges of Multi-task Learning and AdapterFusion discussed in Chapter 2, we hypothesize the following:

1. *Multi-task learning suffers from Catastrophic Interference and does not surpass the CORe approach.*
2. *AdapterFusion mitigates the Catastrophic Interference problem and surpasses the CORe approach.*
3. *Additionally, we expect AdapterFusion to perform better than other approaches in terms of training time and resource usage.*

5.2 Baseline Models

CORe approach introduced by Aken et al., 2021 has surpassed all the other baselines in the past for predicting clinical outcome tasks. In this work, we aim to improve upon the progress made by CORe on the clinical outcome prediction tasks. Therefore, we include CORe as one of our Baseline models. Additionally, similar to the authors of CORe, we include BERT and BioBERT, a BERT's variant pre-trained on biomedical text, into our list of Baseline models. As discussed in section 3.4, we primarily use AUROC (Area under the ROC curve) as our Evaluation metric. For Error Analysis, we use Precision, Recall, and F1 score metrics as well. Table 5.1 shows the performance of the BioBERT Base and CORe on the four clinical outcome tasks. In this work, the prediction performance (AUROC) of the Baseline models have been accessed directly from Aken et al., 2021.

	Diagnosis (1266 classes)	*Procedures* (711 classes)	*In-Hospital Mortality* (2 classes)	*Length-of-Stay* (4 classes)
BERT Base	82.08	85.84	81.13	70.40
BioBERT Base	82.81	86.36	82.55	71.59
CORe	**83.54**	**87.65**	**84.04**	**72.53**

TABLE 5.1: Baseline models, BioBERT and CORe on clinical outcome prediction tasks in macro-averaged AUROC. The CORe approach surpasses BioBERT Base model in all four tasks.(Aken et al., 2021)

5.3 Results

5.3.1 Experimental Setup

For both, Multi-task Learning and AdapterFusion, the following parameters stay constant throughout all of our experiments

- **Batch Size**: Batch size refers to the number of training examples used in one iteration. There has been ongoing research on how batch size affects the training process. However, large batch sizes directly affect the computing resources required to process the data. Hence, following the authors of CORe, we use a small batch size of 20 for all of our experiments.
- **Evaluation steps**: It represents the number of steps after which the model is evaluated on the validation data during the training process. Following CORe approach, we set Evaluation steps to 500.
- **Max Sequence Length**: It represents the maximum length of the sequence a model can take as input. BERT cannot take a sequence more than a length of 512 as inputs Devlin et al., 2019. The sequences with a length of more than 512 are truncated. Therefore, for all our experiments, CORE (a variant of BERT incorporating patient trajectories) has the maximum sequence length set to be 512. For MIMIC III data, 62% of the admission notes have a sequence length > 512.

5.3.2 Multi-task Learning

As discussed in Section 3.3, we introduce tasks to our experiments step by step, which led to a total of eight task settings. Table 5.2 depicts the best results on the

	Experiment Setting	*Diagnosis* (1266 classes)	*Procedures* (711 classes)	*In-Hospital Mortality* (2 classes)	*Length-of-Stay* (4 classes)
Baselines	BioBERT Base	82.81	86.36	82.55	71.59
	CORe	83.54	87.65	**84.04**	72.53
MTL	Diagnosis - Procedures	82.45	**90.50**	-	
	Diagnosis - MP	79.70	-	77.53	-
	Diagnosis - LOS	76.55	-		57.07
	Procedures - MP	-	86.27	74.81	-
	Procedures - LOS	-	87.71	-	63.74
	Diagnosis - Procedures - MP	75.86	85.34	78.46	-
	Diagnosis - Procedures - LOS	77.52	87.12	-	57.51
	Diagnosis - Procedures - MP - LOS	69.54	77.79	75.64	62.98
Adapters	ST-A	**83.98**	87.21	83.15	**75.45**
	AdapterFusion	77.15	86.91	79.30	75.15

TABLE 5.2: Performance of Baseline models (BioBERT and CORe), Traditional Multi-task learning (**MTL**), Single-task Adapters **(ST-A)**, and AdapterFusion on clinical outcome prediction tasks in macro-averaged % AUROC using the **test dataset**. For Multi-task learning, the performance worsens as new tasks are included in the setup. AdapterFusion performs worse than the baselines (except on length of stay task). *Single-task Adapters surpass all experiments (including baselines) for Diagnosis and Length of Stay tasks. ST-As produces results close to the CORe approach on Procedures and Mortality Prediction tasks.*

test dataset in the eight task settings using Traditional Multi-task Learning after Hyperparameter optimization. *The only improvement was observed in the AUROC for Procedures task when trained with Diagnosis task in a multi-task setup.* As we introduced Mortality in the experiment, the AUROC for both Diagnosis and Procedures declined compared to "Diagnosis - Procedures" task setting. Similarly, on including Length of Stay with Diagnosis and Procedures in a multi-task setup, AUROC for both, Diagnosis and Procedures task decreased.

We summarize our observations from the experiments on Multi-task learning approach below:

- **Avoids Overfitting problem:** We observe the performance of Multi-task learning on the validation data as well. It is done to validate if Multi-task learning helps solve the problem of Overfitting. Table 5.3 depicts the best results on the Validation dataset in the eight task settings using Traditional Multi-task Learning. On comparing the results of Multi-task learning experiments on Test and validation data, we do not observe much variance in the % AUROC. Therefore, we conclude that Multi-task learning helps avoid the Overfitting problem posed by CORe, where all four tasks are trained separately.
- **Suffers from Catastrophic Interference problem:** After observing the results in eight different settings with Multi-task learning, we interpreted that our experiments suffered from Catastrophic Interference. Catastrophic Interference is a common challenge faced by Multi-task learning where the deep learning model forgets the old parameters on learning new parameters. For example, in

	Experiment Setting	*Diagnosis* (1266 classes)	*Procedures* (711 classes)	*In-Hospital Mortality* (2 classes)	*Length-of-Stay* (4 classes)
MTL	Diagnosis - Procedures	82.13	**90.09**	-	
	Diagnosis - MP	80.27	-	76.35	-
	Diagnosis - LOS	75.90	-		56.52
	Procedures - MP	-	87.09	74.49	-
	Procedures - LOS	-	88.70	-	63.42
	Diagnosis - Procedures - MP	76.30	86.18	76.37	-
	Diagnosis - Procedures - LOS	76.90	87.71	-	57.19
	Diagnosis - Procedures - MP - LOS	70.80	77.55	75.71	61.67
Adapters	ST-A	84.30	87.22	82.30	**74.50**
	AdapterFusion	**85.26**	89.82	**82.95**	74.45

TABLE 5.3: Performance of Traditional Multi-task learning (**MTL**), Single-task Adapters **(ST-A)** and AdapterFusion on clinical outcome prediction tasks in macro-averaged % AUROC using the **validation dataset**. In comparison with the results on the test dataset (Table 5.2), *we observe an overfitting problem with AdapterFusion approach.*

the "Diagnosis-Procedures-Mortality-Length of Stay" setting, the model tends to worsen the performance on clinical outcome tasks compared to CORe results (where each task is trained separately) due to catastrophic interference.

- **High Training Time:** Moreover, we observe that the average training time for Multi-task learning experiments is 20 hours. "Diagnosis-Procedures-Mortality-Length of Stay" setting includes all four clinical outcome tasks and took the longest training time (24 hours) as well.
- **High Resource Usage:** Additionally, we inspected that the model size using Multi-task learning for each experiment averaged 2 GigaBytes.

To conclude, the Multi-task learning experiments on clinical outcome tasks validated our hypothesis. We observed that Multi-task learning experiments suffer from Catastrophic Interference. Procedures is the only task that improved with Multi-task learning experiment in "Diagnosis-Procedures" task setting.

Hyperparameter Analysis for Multi-task learning experiments

As discussed in the previous section, we perform hyperparameter optimization to find the best values for our model parameters. While evaluating our experiments, we observed that values for "task weights" highly impact the results across all experiments in a consistent manner. Task weights represent the weight each task can contribute towards updating model parameters during the training process. As mentioned in Table 4.1, its value can range from 0.001 to 1. We observed that tasks like Diagnosis and Procedures require more training steps than the Length of Stay and Mortality Prediction tasks. Hence, the "task weights" helped us in balancing the model's performance on multiple tasks.

High task weights with optimal Learning rate work best for Diagnosis and Procedures tasks. While working on our first Multi-task experiment with Diagnosis and Procedures tasks in the setting, we observed that they only work best when the weights for both tasks are set to 1. The combined AUROC for Diagnosis and Procedures peaks only when their weight is set to 1. There were cases when the setup did not perform good even with the task weights set to 1 for Diagnosis and Procedures. Therefore, we observed that finding an optimal value of the learning also creates an high impact on the model's performance.

Hence, an optimal learning rate combined with task weights seems to be a perfect recipe for a model to perform better in Multi-task Learning experiments. We, therefore, always set the task weights for Diagnosis and procedures to 1 and optimize the values for other tasks where necessary.

Low task weights work best for Mortality and Length of Stay tasks. Similarly, we found from our experiments that the model provides the best performance in most cases when task weights for Mortality Prediction and Length of Stay are set under 0.2. As they require less number of training steps compared to the Diagnosis and Procedures tasks, low task weights for Mortality and Length of Stay tasks help in finding the right balance for the model learning process.

5.3.3 AdapterFusion

In this section, we share our results on clinical outcome prediction using AdapterFusion. As discussed in Section 3.3, we use Fusion with Single-task Adapters for our experiments. In this approach, we first fine-tuned Adapter modules for each task separately. In the next step, we use fine-tuned Single Task Adapters for the fusion process. Unlike, Multi-task learning where each task setting had one model, for AdapterFusion, we have four different models for each task.

AdapterFusion surpasses all experiments on Validation data for Diagnosis task. Table 5.3 depicts the performance of ST-A and AdapterFusion on the Validation data. We observe that the performance of ST-As and AdapterFusion is similar for Length of Stay and Mortality prediction tasks. However, AdapterFusion surpasses ST-As on Diagnosis and Procedures task.

AdapterFusion performs worse on Test data vs Validation data. We expected the performance of Single-task Adapters and AdapterFusion on the test dataset to be similar as that on the validation data. However, we found that % AUROC on the Test dataset for AdapterFusion worsened compared to observations on the Validation data. For ST-As, the % AUROC on the test dataset was quite similar to our observations on the validation dataset. Table 5.2 represents the performance of ST-As and AdapterFusion on the test dataset. We also include the results for Baseline models and Multi-task learning experiments for easier comparison.

To summarize, **Single-Task Adapters perform better than the AdapterFusion and on-par with the results from CORe.**

Following are our observations from the experiments on Single-Task Adapters and AdapterFusion in terms of challenges and advantages we discussed in Chapter 1 for AdapterFusion:

- **ST-As avoid overfitting, and AdapterFusion suffers from overfitting problem:** We observe that Single-Task Adapters perform equally well on the Test data as on the Validation data. However, AdapterFusion experiments result in worse performance on the Test Data vs. Validation data. It suggests that AdapterFusion could lead to Overfitting. We analyze the performance of Fusion experiments via Attention plots in Section 5.4.2.
- **ST-As take the lowest training time:** Moreover, we observe that the average training time for Multi-task learning experiments is 20 hours. "Diagnosis-Procedures-Mortality-Length of Stay" setting includes all four clinical outcome tasks and took the longest training time (24 hours) as well.

  On average, the training time for Adapter Modules on the cluster varied between 10 to 14 hours, which is less than the Traditional Multi-task Learning approach. However, we observed that as we included more tasks into the Fusion process, the training time increased exponentially.
- **ST-As have the lowest resource usage:** We inspected that the model size for Single Task Adapters were 5.7 MBs whereas for AdapterFusion models, the size was 82 MBs. Both of them are extremely light-weight as compared to models from Traditional Multi-task learning.

5.3.4 Summary

Lastly, we provide a task-wise summaries comparing all three approaches (Multi-task learning, Single-task Adapters and AdapterFusion) below:

1. ***Length of Stay:*** AdapterFusion and Single Task Adapters produced better results than Traditional Multi-task learning and the original CORe approach.
2. ***Diagnosis:*** The AUROC for Single task Adapters is slightly higher than the CORe model. AdapterFusion produced worse than the baselines on this task.
3. ***Mortality:*** We did not find any improvement in this task either from Traditional Multi-task learning or AdapterFusion. However, the AUROC for ST-As (83.15) is very close to that of the CORe approach (84.04).

4. ***Procedures:*** "Diagnosis-Procedures" setting from Multi-task Learning surpassed all other experiments. Additionally, the AUROC using ST-As (87.21) is close the CORe approach (87.65).

5.4 Error Analysis

The goal of this work was to compare and analyze the performance of Multi-task Learning with AdapterFusion in clinical domain. However, on observing the results (AUROC) from previous section, we believe it is important to also analyze Single-Task Adapters in depth. As Single-task Adapters surpass AdapterFusion results for clinical outcome tasks, we study their results as well in this section.

5.4.1 Quantitative Error Analysis

We analyze the class-wise performance of Multi-task Learning, Single-Task Adapters, and AdapterFusion for each task in this section. Additionally, as we have same tasks in different settings using Multi-task learning, we use the results of a task from the setting that provides its best AUROC (Table 5.2).

Diagnosis

As AUROC is highest for Diagnosis in "Diagnosis-Procedures" setting (out of all Multi-task learning experiments), we use it to represent the performance of Multi-task learning approach. Also, as Diagnosis is a multi-label classification task with more than a thousand classes, we observe the Precision, Recall, and F1 score metrics of select ICD-9 codes which have non-zero prediction probability by the "Diagnosis-Procedures" experiment on test dataset and compare them with the performance of other different modeling approaches.

AdapterFusion has zero prediction probability for 77% of Diagnosis ICD-9 codes. Table 5.4 depicts the Precision, Recall and F1 scores of Traditional Multi-task Learning (MTL), Single-Task Adapters (ST-A), and AdapterFusion on select Diagnosis ICD-9 codes. We noticed that AdapterFusion's inability to correctly predict Diagnosis codes is also seen at the class level. Out of the 9 ICD-9 codes listed, AdapterFusion has zero prediction probability for 7 of them on the test dataset.

ST-As are more precise at predicting Diagnosis ICD-9 codes. From Table 5.2, we understand that the AUROC for Single Task Adapters and "Diagnosis-Procedures" setting are very close. Likewise, on class level, we observed that Single-task Adapters have higher precision for each ICD-9 code when compared to Multi-task learning. It helps us in understanding that with Single-Task Adapters, we would experience less False Positive cases of prediction than with Multi-task Learning approach.

		Precision			Recall			F1		
ICD-9 code	*# Examples*	*MTL*	*ST-A*	*Fusion*	*MTL*	*ST-A*	*Fusion*	*MTL*	*ST-A*	*Fusion*
008	307	0.42	0.82	0.00	0.03	0.05	0.00	0.06	0.09	0.00
0084	282	0.43	0.69	0.00	0.03	0.03	0.00	0.06	0.06	0.00
038	1,251	0.56	0.59	0.51	0.51	0.53	0.43	0.54	0.56	0.47
0381	167	0.26	0.28	0.00	0.08	0.04	0.00	0.13	0.07	0.00
0384	209	0.32	0.44	0.00	0.13	0.02	0.00	0.18	0.04	0.00
0389	742	0.41	0.48	0.44	0.26	0.18	0.12	0.32	0.26	0.19
041	850	0.31	0.45	0.00	0.08	0.02	0.00	0.13	0.03	0.00
0411	238	0.22	0.50	0.00	0.03	0.00	0.00	0.06	0.01	0.00

TABLE 5.4: Precision, Recall, and F1 scores for select Diagnosis ICD-9 codes. AdapterFusion has zero prediction probability for 77% of the analyzed codes. ST-As have higher precision score compared to the Multi-task learning (MTL) experiment. We use "Diagnosis-Procedures" task setting to represent the results for MTL in this table.

Procedures

Similar to Diagnosis, we observe the performance of Multi-task Learning, Single-task Adapters, and AdapterFusion on select ICD-9 Procedure codes which have non-zero prediction probability on test dataset via Multi-task learning experiment. Once again, we choose "Diagnosis-Procedure" task setting to represent the results for Multi-task learning approach as Procedures task performed the best in that setup (highest AUROC across all experiments).

Multi-task learning performs best across the sample Procedure ICD-9 codes. Table 5.5 represents the performance of the three approaches on select Procedures codes. As expected, we notice that there are few codes (001,0017,0014) that Multi-task learning approach can predict but Single-task Adapters and AdapterFusion have zero prediction probability. Additionally, we noticed that on average Single-Task Adapters have higher Precision than the other two approaches, but it but leads to a slightly lower Recall across all the codes.

		Precision			Recall			F1		
ICD-9 code	*# Examples*	*MTL*	*ST-A*	*Fusion*	*MTL*	*ST-A*	*Fusion*	*MTL*	*ST-A*	*Fusion*
004	399	0.65	0.70	0.68	0.51	0.42	0.29	0.57	0.53	0.41
001	396	0.20	0.00	0.00	0.03	0.00	0.00	0.05	0.00	0.00
0040	305	0.62	0.61	0.59	0.46	0.37	0.18	0.53	0.46	0.28
006	291	0.68	0.71	0.72	0.63	0.55	0.33	0.66	0.62	0.45
0066	260	0.66	0.69	0.70	0.65	0.59	0.30	0.65	0.63	0.42
0045	178	0.55	0.50	1.00	0.31	0.19	0.01	0.40	0.27	0.01
0017	160	0.13	0.00	0.00	0.01	0.00	0.00	0.02	0.00	0.00
0014	149	0.04	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00
015	131	0.71	0.77	0.87	0.64	0.62	0.25	0.67	0.69	0.39
013	126	0.53	0.00	0.00	0.48	0.00	0.00	0.50	0.00	0.00

TABLE 5.5: Precision, Recall, and F1 scores for select Procedures ICD-9 codes.We use "Diagnosis-Procedures" task setting to represent the results for MTL in this table. MTL performs best across all ICD-9 codes with high F1 score compared to Single-task Adapters and AdapterFusion.

Mortality

For Mortality Prediction, we use "Diagnosis-Procedures-Mortality" setting to represent the results of Multi-task learning approach. Table 5.6 outlines the class-wise performance of the three approaches on the Mortality Prediction task. "0" and "1" Labels represent the instances when the Patient did not die and when the Patient died during the hospitalization, respectively.

ST-A has higher precision in correctly predicting the Mortality of a patient. Like Procedures and Diagnosis, we noticed a higher Precision of Single-task Adapters than the other two approaches for Label "1" with a slightly lower Recall. On the other hand, AdapterFusion with higher Recall leads to fewer False Negative predictions for Label "1".

		Precision			Recall			F1		
Label	*# Examples*	*MTL*	*ST-A*	*Fusion*	*MTL*	*ST-A*	*Fusion*	*MTL*	*ST-A*	*Fusion*
0	8,797	0.92	0.92	0.95	0.96	0.97	0.82	0.94	0.94	0.88
1	1,033	0.46	0.54	0.28	0.28	0.26	0.61	0.35	0.35	0.38

TABLE 5.6: Precision, Recall, and F1 scores for Mortality task labels. "0" Label represents that the patient did not die during the hospitalization and "1" represents that the patient passed away during the hospitalization. On average, Single-task Adapters have higher precision in predicting if the patient would die during the hospitalization. We use "Diagnosis-Procedures-Mortality" task setting to represent the results for MTL in this table.

Length of Stay

In the case of the Length of Stay task, we use "Procedures-Length of Stay" task to represent the Multi-task learning approach. Table 5.7, shows the results of the three approaches on this task. To be precise, Label "0" represents the patient's stay at the hospital for less than or equal to 3 days, "1" represents the stay between 4 and 7 days, "2" represents the stay between 8 and 14 days, and "3" represents a stay of more than 14 days.

ST-A and AdapterFusion perform similar in predicting Length of Stay across all class labels. As expected, the class-wise performance of Single-task Adapters and AdapterFusion are similar and better than the Multi-task learning experiment. Additionally, we notice that because due to less class variance for this task, the performance is similar of a model across different classes. However, Label "1" (Patient's stay between 4 and 7 days) observes the highest F1 score using all approaches. We believe it is due to a high frequency of Label "1" in the data.

		Precision			Recall			F1		
Label	*# Examples*	*MTL*	*ST-A*	*Fusion*	*MTL*	*ST-A*	*Fusion*	*MTL*	*ST-A*	*Fusion*
0	1,121	0.36	0.47	0.46	0.42	0.34	0.35	0.39	0.40	0.40
1	3,328	0.43	0.47	0.48	0.51	0.50	0.53	0.47	0.49	0.50
2	2,692	0.36	0.38	0.37	0.41	0.36	0.39	0.38	0.37	0.38
3	1,656	0.33	0.36	0.39	0.31	0.41	0.36	0.32	0.38	0.37

TABLE 5.7: Precision, Recall, and F1 scores for select Mortality Prediction task labels. Label "0" : <3, "1" : 4-7, "2": 8-14, "3": >14 days at the hospital. Single-task Adapters and AdapterFusion have very similar scores across all labels and perform better than the Multi-task learning approach. We use "Procedures-Length of Stay" task setting to represent the results for MTL in this table.

5.4.2 Qualitative Error Analysis

Attention Plots for AdapterFusion

In order to better understand the workings of AdapterFusion and its worse performance on the test dataset than the validation dataset, we analyze the attention weights in the AdapterFusion layer, similar to the authors of AdapterFusion Pfeiffer et al., 2021. Figure 5.1 visualizes the weights of the AdapterFusion layer for all Single-Task Adapters. The rows represent the target task and the columns represent the pre-trained Single-Task Adapters. The higher the weight of a Single-Task Adapter (m) for a Target Task (n), Pfeiffer et al., 2021 assumes that the information provided by Adapter m is more beneficial for Target task, n.

We list our insights for the Fusion target tasks using the Attention plots below:

1. **Diagnosis has high activation for Procedures and Mortality task adapters** Diagnosis as a task pays more relevance towards Procedures and Mortality tasks across all layers. In Layer 4, its activation weight is highest for the Procedures task and close to zero for the other tasks. Similarly, in the last layer (Layer 12), its activation weight is highest for the Mortality task. The only layers where the activations are high for its own adapter are layer 3 and 8, that too, only slightly.

2. **Procedures has high activation weight for Mortality and its own adapters** Similar to the AdapterFusion process with Diagnosis task, we observe that Procedures also pay high relevance to the Mortality prediction task. However, unlike Diagnosis, Procedures do give high activation weight to its own adapter. It also explains the lower performance of Diagnosis than Procedures on the Test data using AdapterFusion.

3. **Mortality has high activation for Procedures task** Mortality task's activation weights are primarily high for the Procedures task across all layers. It also gives relevance to the adapters of other tasks, but the weights are very low.

4. **Length of Stay only gives high activation weight to its own adapter** Unlike all other three tasks, Length of Stay provides relevance to its own adapter.

This also helps us understand why the performance of the other three tasks decreased on the test data compared to the validation data, but that did not happen for the Length of Stay task.

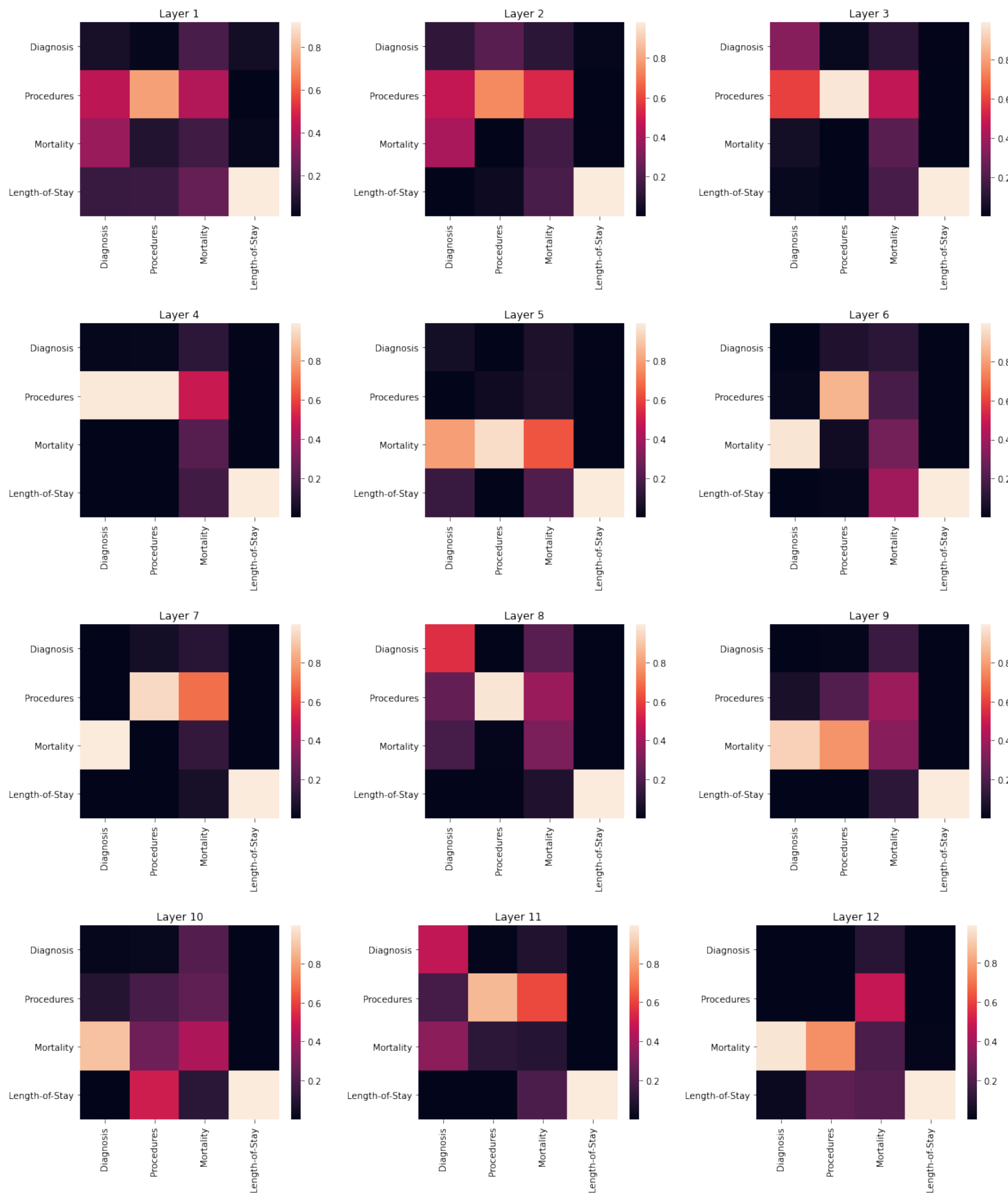


FIGURE 5.1: AdapterFusion activations of pretrained Single task Adapters. The rows represent the target task and the columns represent the related task. Diagnosis task has high activation weight for Procedures and Mortality tasks across all layers. Procedures task has high activation for Mortality and its own adapter. Similarly, Mortality task gives high relevance to Procedures and its own adapter. Unlike other tasks, Length of Stay gives high relevance just to its own adapter across all layers. This analysis is inspired by the work performed by the authors of AdapterFusion in Pfeiffer et al., 2021.

Hypothesis:
High class variability could lead to poor performance for AdapterFusion. After analyzing the AdapterFusion activation of the four tasks, we assume that the high variability of class labels amongst the tasks led to their worse performance on the Test Dataset vs Validation Dataset. As discussed in Section 3.2.1, Diagnosis, Procedures and Mortality tasks are highly imbalanced as compared to the Length of Stay.

Validation:
AdapterFusion's performance on Diagnosis improved upon using less ST-As. We learnt from Traditional Multi-task learning experiments with "Diagnosis-Procedures" task setting that the two tasks help improve each other's performance, especially Procedures task for which we observed 90.09 % AUROC (highest of all experiments, Table 5.2).
Therefore, we decided to use this information when working with AdapterFusion experiments. We used just Diagnosis and Procedures task adapters to improve on the Diagnosis task using the Fusion process. We observed an improvement on the Diagnosis task with an AUROC of 77.15 (including adapters for all four tasks) to 79.57 (including adapter for just Diagnosis and procedures). This led us to understand that the adapters from related tasks are not helping to improve the Diagnosis task using fusion.
Hence, we hypothesize that during the Fusion process, the models learned wrong contextual representation by activating adapters for other tasks (except for Length of Stay). However, we believe further research is required to validate our hypothesis.

5.4.3 Analysis of Diagnosis codes

As mentioned in Chapter 1, this goal's objective is to study the improvements in Diagnosis predictions. Therefore, we aimed to analyze the quality of predicted Diagnosis codes for 20 sample Admission notes, which were also used by Aken et al., 2021 in their work. As Diagnosis is an extreme multi-label classification task, it is very challenging to analyze multiple labels for different approaches. We also took into consideration the insights shared by the authors of CORe on the incomplete labeling of the ICD-9 codes. The authors revealed that 60% of the samples they analyzed were partially under-coded, meaning that MIMIC III dataset did not include all of the required ICD-9 codes for the Diagnosis or Procedure. Hence, we do not specifically analyze the False Positives and False Negatives in this work.

Nevertheless, we decided to take the MIMIC III ICD-9 codes as the Ground Truth and checked if the corresponding ICD-9 codes were predicted by Multi-task learning (Diagnosis-Procedures setting), Single-Task Adapters, and AdapterFusion.

On average, for an Admission note, the number of Diagnosis ICD-9 codes were 65. The maximum number of ICD-9 codes for a note were 120.

AdapterFusion provides high relevance to "Hypertension" Diagnosis codes. For AdapterFusion, we noticed that for 95% of the samples, the model predicted "401" and "4019" ICD-9 codes which represent *Hypertension*. On Average, the model predicted just 5 Diagnosis code for an Admission Note. The poor performance of the AdapterFusion model was expected from low AUROC score but this analysis helped us understand its predictions on a deeper level. We believe the reason for the model's focus on Hypertension could also be because they are one of the top 10 most frequent ICD-9 codes in the MIMIC III dataset.
Given AdapterFusion's poor quality of results on Diagnosis task and the ability of predict only a few Diagnosis codes, the average match rate between the MIMIC III labels and the codes predicted by AdapterFusion was 5%.

Multi-task Learning predictions has a match rate of 33% with MIMIC III labels. On average, the Traditional Multi-task approach (Daignosis-Procedures) predicted 20 Diagnosis ICD-9 codes. The maximum number of codes predicted were 42. The average match rate between the labels predicted by Multi-task learning model and MIMIC III labels was 33%. On a closer evaluation, we noticed that the model was able to predict the major Diagnosis ICD-9 codes (3 digit) for most samples, however, it failed in predicting the ICD-9 codes with granular information (greater than 3 digits). For instance, it predicts 272 ("Disorders of lipoid metabolism") ICD-9 code for a note but misses 2720 ("Pure hypercholesterolemia"), which was also present in the note as text under "Medical History" section.

On average, ST-A predictions has a match rate of 38% with MIMIC III Diagnosis codes. In the previous section, we presented that Single-Task Adapters surpassed all or experiments and the Baselines (CORE and BioBERT) on Diagnosis task. In this section, we further analyze the quality of its predictions.

On average, the ST-A predicted 32 Diagnosis ICD-9 codes. The maximum number of codes predicted were 62, much higher than the Multi-task learning model. The average match rate between the labels predicted by ST-A and MIMIC III labels was 38%. Unlike the Multi-task learning model, we noticed that the Adapter model was able to predict ICD-9 codes with granular information more frequently. Taking the previous example we discussed for Multi-task learning approach, the ST-A predicted ICD-9 code 2720 in addition to 272.

5.5 Discussion

As mentioned in Section 5.1, at the beginning of this work, we expected to observe better performance on clinical outcome tasks from AdapterFusion compared to Traditional Multi-task learning. However, as we progressed with our experiments, we observed Single-Task Adapters surpassed the results of Multi-task learning and AdapterFusion process and resolved the challenges this work aimed to solve.

In the following sections, we summarize the results of our experiments for each problem we target to solve via this work.

ST-As use the lowest amount of training time and resources. We noted that the Training time for each Multi-task learning experiment took approximately 20 hours. As we include more and more tasks into the multi-task setup the training time increases. For Single-task Adapters and AdapterFusion, the average training time varied from 10 to 14 hours.

The Single-Task Adapter models use the lowest amount of storage space, i.e. 5.7 MBs. Collectively, separate ST-A model for each task would take up 24 MBs of total space. For a single AdapterFusion model, the storage space is 82 MBs. A Fusion model for all tasks collectively takes 328 MBs. Unlike, ST-A and Fusion models, Traditional Multi-task learning models using FARM takes up 2GBs of space.

AdapterFusion experiences overfitting problem. We observed Overfitting problem only with the AdapterFusion models. It provided low AUROC on the test data as compared to the validation data for Diagnosis, Procedures and Mortality tasks. For Length of Stay, we observed approximately the same AUROC on the test data as that of the validation data. We hypothesize that this is due to the high class variability among Diagnosis, Procedures, and Mortality tasks. Additionally, on analyzing the attention plots of the AdapterFusion models, we discovered that all three tasks pay more relevance to the adapters of other tasks. On the other hand, during the fusion process, the Length of Stay model only gives high attention to its own adapters across all layers, which helps it perform better than the other tasks.

Only Procedures task gain information using inter-contextual representations. One of the major goal of this work was to understand if through Multi-task learning approaches or AdapterFusion we can understand the inter-contextual representations of the different tasks. Through our experiments with Traditional Multi-task learning, the only improvement we observed compared to Baselines was for Procedures task in "Diagnosis-Procedures" task. This helped us understand that Procedures task gained information from the Diagnosis task via the multi-task setup.

Nevertheless, for all other seven Multi-task learning experiments, we observed poor results for all tasks. The performance of the models kept on worsening as we included more and more tasks into the setup. It reveals that the only task that gain information in multi-task setup is Procedures from Diagnosis task. We experienced poor performance of the models on adding Mortality and Length of Stay task to the experimental settings.

AdapterFusion doesn't solve catastrophic interference problem in clinical domain. Our motivation to include AdapterFusion in this work was to validate if it helps us solve the problem of Catastrophic Interference posed by Multi-task Learning. However, as discussed in 5.5, AdapterFusion resulted in poor results on the Diagnosis, Procedures, and Mortality tasks and did not help us in solving this problem.

To summarize, we observed best performance on Procedures task via Multi-task Learning. However, all other experiments with Multi-task learning produced much worse results than the Baselines. Likewise, AdapterFusion did not validate the hypothesis of this work. On the other hand, Single Task Adapters outperformed other modeling approaches as per our observations in this work. They are extremely lightweight models with low training time. It results in faster inference which is good for hospitals with low compute resources.

5.6 Summary

In this Chapter, we first explained this work's hypothesis which was AdapterFusion surpassing the Baselines and other experiments on the clinical outcome tasks, on the basis of results presented the authors of AdapterFusion in Pfeiffer et al., 2021. We presented the results of Baseline models, namely BERT Base, BioBERT, and CORe on the four clinical outcome tasks. Next, we presented the results of our Multi-task learning experiments and compared them with the Single-Task Adapters, and AdapterFusion , on both the Validation data and the test data. We explain the Catastrophic Interference problem observed in the Multi-task learning experiments except for "Diagnosis-Procedures" setting, where the AUROC for Procedures surpassed that of the Baselines as well as our other experiments. It led us to understand that the only related tasks out of the four are Diagnosis and Procedures. Additionally, we discuss in detail our insights from the Hyperparameter optimization for the multi-task learning experiments. We observed that the Task weights were highly effective in tuning the models. For AdapterFusion, we observed overfitting problem and analyzed the attention weights of the fusion models to understand its poor performance on the tasks. We also study the class-wise performance of the different learning algorithms on the four clinical outcome tasks. As the primary task of this work was to improve the results on the Diagnosis task, we analyzed the quality of the predictions of Multi-task Learning, ST-A and Fusion process for Diagnosis ICD-9 codes. Finally, we rephrase our findings in the Discussion section.

Chapter 6

Conclusion and Future Work

6.1 Summary

In this work, we aimed to improve upon the advancements made by the CORe approach at predicting clinical outcome tasks. The authors of CORe proposed a novel task setup in which they use Admission notes of a patient to predict four discharge tasks, namely, Diagnosis, Procedures, Mortality, and Length of Stay of the patient in the hospital. The knowledge of these outcomes at the patient's admission time can help doctors not overlook risks and better planning the resources of the hospital. In the CORe approach, the authors trained four separate models for each task, leading to problems like overfitting, high training time, and resource usage. Primarily, by training separate models, it is impossible to understand if different tasks can gain knowledge from each other and help in the better performance of a model.

Methodology and Implementation We experiment with Traditional Multi-task learning and AdapterFusion to resolve the challenges posed by the CORe approach. AdapterFusion is a novel methodology proposed by Pfeiffer et al., 2021 to overcome the problem of catastrophic interference posed by Traditional Multi-task learning. We use Fusion with Single task Adapters in this work. Adapter modules introduce only a few trainable parameters per task (less than traditional fine-tuned models), leading to high performance and low training time for the target task. We apply the Fusion process to four separate single task adapters for our experiments. We use the FARM library to implement Traditional Multi-task learning experiments and newly built adapter-transformers for the Single task Adapters and AdapterFusion approach. Similar to the authors of CORe, we used MIMIC III, a publicly available dataset including patient admission notes.

Evaluation We hypothesized that Multi-task learning suffers from catastrophic interference and does not surpass the performance of CORe. Additionally, we hypothesized that AdapterFusion solves catastrophic interference and overperforms the CORe approach. During the evaluation process, our hypothesis was validated for the Multi-task learning approach. The only task that gained knowledge and improved its performance via multi-task learning was the Procedures task. As we included more and more tasks in the setup, the models faced catastrophic interference. However, our hypothesis was proved wrong for the AdapterFusion process in the clinical domain. We observed that AdapterFusion suffered from an overfitting problem for the Diagnosis, Procedures, and Mortality Prediction task. Based on the activation weights we studied in the Error Analysis section, we assume the poor performance of AdapterFusion is caused by high class variability amongst the tasks. On the other hand, we observed that the Single-task Adapters surpassed baselines and all other experiments on Diagnosis and Length of Stay tasks. Their performance on Procedures and Mortality tasks was also on par with the CORe approach.

Conclusion As per our experiments, the Single task Adapters worked best for predicting Diagnosis and Length of stay tasks. Additionally, their AUROC for Procedures and Mortality tasks is close to that of the CORe approach. As very few parameters are updated during the training process for Adapters, it is not susceptible to an overfitting problem, unlike AdapterFusion. Moreover, the training time and the resources used by Single task Adapters are the lowest compared to all other experiments, including the Baselines. Furthermore, during the error analysis process, we noticed that Single-task Adapters are more precise with their predictions across all tasks.

6.2 Future Work

Although Single-task Adapters showed promising results and outperformed the Baselines on Diagnosis and Length of Stay tasks, there is still room for improvement. We list below four approaches through which we can further improve the prediction results on clinical outcome tasks.

Additional data sources We believe to further improve the performance on clinical outcome prediction tasks by including information about the clinical domain via additional datasets. Our idea is to include additional non-hospital data sources and knowledge bases using, say, image data. We assume that one reason for not observing improvement in Diagnosis, Mortality, and Length of Stay tasks via Multi-task learning was that the input data for all tasks was the same, and the model did not find additional information to gain from the other tasks. The multimodal dataset approach can help rectify the problem.

AdapterDrop In Rücklé et al., 2020, the authors propose Adapter Drop where they drop adapters from specific layers, improving the performance on the task as well as reducing the inference time by 30%. Dropping particular layers from our AdapterFusion experiments may help improve its performance on the clinical outcome tasks.

Hyperformer In Mahabadi et al., 2021, the authors propose a hyperformer to share adapter parameters across each task in a multi-task setup. They present that we can learn adapter parameters across all layers via a hypernetwork depending on the task, adapter position, and layer id in a transformer model. The authors also present that this architecture reduces catastrophic inference, a common problem with Multi-task learning.

Bibliography

Aken, Betty van et al. (Apr. 2021). “Clinical Outcome Prediction from Admission Notes using Self-Supervised Knowledge Integration”. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, pp. 881–893. URL: `https://aclanthology.org/2021.eacl-main.75`.

Alammar, Jay (2018). “The Illustrated BERT, ELMo, and co.” In: URL: `https://jalammar.github.io/illustrated-bert/` (visited on 05/01/2021).

Anello, Eugenia (2021). “How to evaluate you model using the Confusion Matrix”. In: URL: `https://towardsai.net/p/data-science/how-to-evaluate-you-model-using-the-confusion-matrix`.

Brownlee, Jason (2017). “Deep Learning with Time Series Forecasting”. In: URL: `https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/` (visited on 07/14/2021).

Devlin, Jacob et al. (June 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: `10.18653/v1/N19-1423`. URL: `https://aclanthology.org/N19-1423`.

Draelos, Rachel Lea Ballantyne (2019). “Measuring Performance: AUC (AUROC)”. In: URL: `https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/`.

Houlsby, Neil et al. (2019). “Parameter-Efficient Transfer Learning for NLP”. In: *Proceedings of the 36th International Conference on Machine Learning*. PMLR, pp. 2790–2799. URL: `http://proceedings.mlr.press/v97/houlsby19a.html`.

Howard, Jeremy and Sebastian Ruder (July 2018). "Universal Language Model Fine-tuning for Text Classification". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 328–339. DOI: 10.18653/v1/P18-1031. URL: https://aclanthology.org/P18-1031.

Irving, Greg et al. (2017). "International variations in primary care physician consultation time: a systematic review of 67 countries". In: *BMJ Open* 7.10. ISSN: 2044-6055. DOI: 10.1136/bmjopen-2017-017902. eprint: https://bmjopen.bmj.com/content/7/10/e017902.full.pdf. URL: https://bmjopen.bmj.com/content/7/10/e017902.

Johnson, A., T. Pollard, and L. Shen (2016). "MIMIC-III, a freely accessible critical care database". In: URL: https://doi.org/10.1038/sdata.2016.35.

Lee, Jinhyuk et al. (Sept. 2019). "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics*. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz682. URL: https://doi.org/10.1093/bioinformatics/btz682.

Lee, Joon et al. (2011). "Open-access MIMIC-II database for intensive care research". In: *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 8315–8318. DOI: 10.1109/IEMBS.2011.6092050.

Li, Liam et al. (2018). "Massively Parallel Hyperparameter Tuning". In: *ArXiv* abs/1810.05934.

Mahabadi, Rabeeh Karimi et al. (2021). "Parameter-efficient Multi-task Fine-tuning for Transformers via Shared Hypernetworks". In: *ArXiv* abs/2106.04489.

McCloskey, Michael and Neal J. Cohen (1989). "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem". English (US). In: *Psychology of Learning and Motivation - Advances in Research and Theory* 24, pp. 109–165. ISSN: 0079-7421. DOI: 10.1016/S0079-7421(08)60536-8.

Nayak, Pandu (2019). "Understanding searches better than ever before". In: URL: https://blog.google/products/search/search-language-understanding-bert/.

Peters, Matthew E. et al. (June 2018). "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

New Orleans, Louisiana: Association for Computational Linguistics, pp. 2227–2237. DOI: `10.18653/v1/N18-1202`. URL: `https://aclanthology.org/N18-1202`.

Pfeiffer, Jonas et al. (2021). "AdapterFusion: Non-Destructive Task Composition for Transfer Learning". In: *EACL*.

Rosenthal, Sara, Ken Barker, and Zhicheng Liang (Nov. 2019). "Leveraging Medical Literature for Section Prediction in Electronic Health Records". 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)*. Hong Kong, China: Association for Computational Linguistics, pp. 4864–4873. DOI: `10.18653/v1/D19-1492`. URL: `https://aclanthology.org/D19-1492`.

Rücklé, Andreas et al. (2020). "AdapterDrop: On the Efficiency of Adapters in Transformers". In: *arXiv*. URL: `https://arxiv.org/abs/2010.11918`.

Ruder, Sebastian (2017). "An Overview of Multi-Task Learning in Deep Neural Networks". In: URL: `https://arxiv.org/abs/1706.05098v1`.

– (2019). *Neural Transfer Learning for Natural Language Processing*. National University of Galway, Ireland.

Sebastiano Barbieri James Kemp, Oscar Perez-Concha Sradha Kotwal Martin Gallagher Angus Ritchie Louisa Jorm (2020). "Benchmarking Deep Learning Architectures for Predicting Readmission to the ICU and Describing Patients-at-Risk". In: DOI: `https://doi.org/10.1038/s41598-020-58053-z`.

Si, Yuqi and Kirk Roberts (2019). "Deep Patient Representation of Clinical Notes via Multi-Task Learning for Mortality Prediction". In: vol. 2019. American Medical Informatics Association, p. 779.

Singh, Hardeep, Ashley N D Meyer, and Eric J Thomas (2014). "The frequency of diagnostic errors in outpatient care: estimations from three large observational studies involving US adult populations". In: *BMJ Quality & Safety* 23.9, pp. 727–731. ISSN: 2044-5415. DOI: `10.1136/bmjqs-2013-002627`. eprint: `https://qualitysafety.bmj.com/content/23/9/727.full.pdf`. URL: `https://qualitysafety.bmj.com/content/23/9/727`.

Topol, Eric (2019). *Deep Medicine: How Artificial Intelligence Can Make Healthcare Human Again*. 1st. USA: Basic Books, Inc. ISBN: 1541644638.

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *ArXiv* abs/1706.03762.

WHO (1975). "International classification of diseases : ninth revision, basic tabulation list with alphabetic index." In: URL: `https://apps.who.int/iris/handle/10665/39473`.