Hierarchical Neural Architectures for Classifying Cancer Pathology Reports

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Shang Gao

December 2019

© by Shang Gao, 2019
All Rights Reserved.

This work is dedicated to Nala, my landlord's cat, who thinks she is the queen of the universe.

Acknowledgments

This work was made possible thanks to the help of my esteemed advisors, Arvind Ramanathan and Georgia Tourassi, and my committee members, Hairong Qi and Russell Zaretzki. I would also like to thank the members of the Biomedical Sciences, Engineering, and Computing group at Oak Ridge National Laboratory for their help with this work – Mohammed Alawad, John Qiu, Jacob Hinkle, Noah Schaefferkoetter, Hong-Jun Yoon, and Blair Christian. Lastly, I would like to thank my wife Dasha Herrmannova for her support and for proofreading this work.

This work has been supported in part by the Joint Design of Advanced Computing Solutions for Cancer (JDACS4C) program established by the U.S. Department of Energy (DOE) and the National Cancer Institute (NCI) of the National Institutes of Health. This work was performed under the auspices of the U.S. Department of Energy by Argonne National Laboratory under Contract DE-AC02-06-CH11357, Lawrence Livermore National Laboratory under Contract DEAC52-07NA27344, Los Alamos National Laboratory under Contract DE-AC5206NA25396, and Oak Ridge National Laboratory under Contract DE-AC05-00OR22725 This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Abstract

Electronic health records (EHRs) are the primary method for documenting and storing patient outcomes in modern healthcare; data mining and machine learning approaches utilize the information stored in EHRs to assist in clinical decision support and other critical healthcare applications. Important information in EHRs is often stored in the form of unstructured clinical text. Unfortunately, the state-of-the-art methods used to automatically extract useful information from unstructured clinical text lags significantly behind the state-of-the-art methods used in the general natural language processing (NLP) community for other tasks such as machine translation, question answering, and sentiment analysis. In this work, we attempt to bridge this gap by applying and developing hierarchical neural approaches to classify key data elements in cancer pathology reports, such as cancer site, histology, grade, and behavior. We (1) show that a hierarchical attention network (HAN), which has strong performance on classifying general text such as Yelp reviews and news snippets, achieves better classification accuracy and macro F-score on identifying cancer site and grade than previous state-of-the-art approaches, (2) develop a novel hierarchical self-attention network (HiSAN) which not only achieves better accuracy and macro F-score in cancer pathology pathology report classification than the HAN but also trains over 10x faster, and (3) introduce a hierarchical framework for incorporating case-level context when classifying cancer pathology reports and show that it gives a significant boost in accuracy and macro F-score.

Table of Contents

1 Introduction **1**

1.1 State of the Art . . . 4

1.1.1 General NLP . . . 4

1.1.2 Clinical NLP . . . 6

1.1.3 Classifying Cancer Pathology Reports . . . 7

1.2 Research Objectives and Approach . . . 10

1.2.1 Hierarchical Attention Networks . . . 12

1.2.2 Hierarchical Self-Attention Networks . . . 14

1.2.3 Capturing Case-Level Context . . . 16

1.3 Dissertation Outline . . . 18

2 Hierarchical Attention Networks for Information Extraction from Cancer Pathology Reports **20**

2.1 Abstract . . . 21

2.2 Objective . . . 22

2.3 Background . . . 23

2.4 Materials and Methods . . . 26

2.4.1 Dataset Description and Preprocessing . . . 26

2.4.2 Word Embeddings . . . 28

2.4.3 Hierarchical Attention Network . . . 28

2.4.4 Hyperparameter Optimization . . . 31

2.4.5 Unsupervised Pretraining Step . . . 31

2.4.6 Experimental Setup and Evaluation Metrics . . . 32

2.4.7 Model Baseline Comparisons . . . 33
2.5 Experimental Results . . . 35
2.5.1 Primary Site Classification . . . 35
2.5.2 Histological Grade Classification . . . 35
2.5.3 Model Visualizations . . . 37
2.5.4 Error Analysis . . . 37
2.6 Discussion and Conclusions . . . 40

3 Classifying Cancer Pathology Reports with Hierarchical Self-Attention Networks 45

3.1 Abstract . . . 46
3.2 Introduction . . . 47
3.3 Background . . . 48
3.4 Materials and Methods . . . 51
3.4.1 Dataset Details . . . 51
3.4.2 Hierarchical Self-Attention Networks . . . 53
3.5 Results . . . 58
3.5.1 Baselines, Hyperparameters, and Setup Details . . . 59
3.5.2 Evaluation Metrics . . . 60
3.5.3 Experimental Results . . . 62
3.6 Discussion . . . 66
3.6.1 Error Analysis . . . 66
3.6.2 Visualizing Document Embeddings . . . 68
3.6.3 Visualizing Word Importance . . . 68
3.7 Conclusion . . . 72

4 Using Case-Level Context to Classify Cancer Pathology Reports 74

4.1 Abstract . . . 75
4.2 Introduction . . . 75
4.3 Background . . . 77
4.4 Materials and Methods . . . 78

4.4.1 Problem Description . . . 78
4.4.2 Capturing Case-Level Context . . . 79
4.4.3 Dataset . . . 86
4.4.4 Baseline Models . . . 87
4.5 Experiments . . . 88
4.5.1 Setup Details . . . 88
4.5.2 Evaluation Metrics . . . 89
4.5.3 Results . . . 90
4.6 Discussion . . . 93
4.7 Conclusion . . . 98

5 Conclusion **100**
5.1 Summary of Results . . . 100
5.2 Impact and Implications . . . 103
5.3 Transfer Learning . . . 105
5.4 Applications to General Text . . . 109
5.5 A Note on ELMo and BERT . . . 113
5.6 Additional Future Work . . . 116

Bibliography **118**

Appendices **137**
A Chapter 2 Supporting Information . . . 138
A.1 Preprocessing Pipeline . . . 138
A.2 Preprocessing Procedures . . . 138
A.3 Corpus Histogram . . . 139
A.4 LSTMs and GRUs . . . 140
A.5 Attention Mechanism . . . 141
A.6 Hyperparameter Optimization Setup . . . 142
A.7 TF-IDF Weighted Word Embeddings . . . 143
A.8 Bootstrapping Procedures for Metrics . . . 143

A.9 Doc2Vec Embeddings . . . 144
A.10 CNN Embeddings . . . 145
A.11 Most Important Words for Each Classification Task . . . 145
A.12 Model Prediction Time . . . 146
A.13 Confusion Matrices . . . 146
A.14 Pubmed Abstract Classification Experiment . . . 150
B Chapter 3 Supporting Information . . . 154
B.1 Dataset Distribution and Label Descriptions . . . 154
B.2 Text Preprocessing and Bootstrapping Details . . . 156
B.3 Classifier F-Score Breakdown by Class . . . 159
B.4 McNemar's Tests of Significance . . . 164
B.5 Classifier Performance on Only Tumor IDs Associated with One Pathology Report . . . 165
B.6 Effect of Attention Heads on HiSAN Performance . . . 165
B.7 Ablation Study of HCAN vs HiSAN . . . 167
B.8 Experiments on Vocabulary Size . . . 167
C Chapter 4 Supporting Information . . . 168
C.1 Dataset Distributions . . . 168
C.2 Detailed Experimental Procedures . . . 171
C.3 Modular vs End-to-End Training . . . 174

Vita **175**

List of Tables

2.1 Distribution of labeled reports for the primary site classification task and the histological grade classification task. 27

2.2 Final test performance of classification models on each classification task. Classifier performance and confidence intervals on individual tasks are shown within parentheses. 36

3.1 Top: test set accuracy and macro F-score (with 95% confidence intervals) on each classification task. Bottom: train and inference times (in seconds) for each classification method; train times for deep learning approaches are displayed as time per epoch × number of epochs to converge. 63

3.2 Test set accuracy and macro F-score (with 95% confidence intervals) on the site and histology tasks using a reduced dataset in which cases from the 15 most prevalent cancer sites are removed. 63

4.1 Accuracy and Macro F-Score (with 95% confidence intervals) of our different methods to capture case-level context on six different classification tasks using the CNN as the baseline. The top row is our baseline without any report level context, the middle group shows results of methods than can access both future and previous reports in a sequence, and the bottom group show results of methods that can only access previous reports in a sequence. 91

4.2 Accuracy and Macro F-Score (with 95% confidence intervals) of our different methods to capture case-level context on six different classification tasks using the HiSAN as the baseline. The top row is our baseline without any report level context, the middle group shows results of methods than can access both future and previous reports in a sequence, and the bottom group show results of methods that can only access previous reports in a sequence. 92

5.1 Dataset descriptions for our transfer learning experiments. We split our full LA + KY dataset in two ways: (1) by registry and (2) top 15 most prevalent cancer sites vs minority cancer sites (all other sites). For each dataset, we show the number of samples in our train, validation, and test sets as well as the number of classes per task. 107

5.2 Accuracy and Macro F-Score (with 95% confidence intervals) of the HiSAN on the LA dataset using different training regiments. 107

5.3 Accuracy and Macro F-Score (with 95% confidence intervals) of the HiSAN on the KY dataset using different training and transfer learning regiments. . 107

5.4 Accuracy and Macro F-Score (with 95% confidence intervals) of the HiSAN on only the minority cancer sites (cancer sites excluding the top 15 most common cancer sites) from the LA+KY dataset. We compare performance using different training and transfer learning regiments. 109

5.5 Dataset Descriptions 111

5.6 Test set accuracy, mean training time for a single document, and total training time on each task. 112

5.7 Validation and Test Accuracy of different classification models on the Yelp '14 sentiment analysis task. All results other than HCAN are obtained from [2]. 115

A.1 Optimized hyperparameters for HAN for each classification task, discovered through gradient boosted tree-based search. We note that for the hyperparameters Number of RNN units per hierarchy and Size of attention hidden layer, the search space was not expanded above 200 and 300 respectively because in both cases the best performing hyperparameter was below the upper limit and was simply rounded up to the nearest 50. 142
A.2 The most important words for each classification task, based on the average word-level attention scores assigned to each word 145
A.3 The average time (in ms) required to classify a single pathology report for each trained classifier. All models were tested using on a single Intel Core i7-3770 CPU @ 3.40GHz. We note that HANs train faster than regular RNNs because whereas RNNs read the entire document as one long sequence, HANS break documents down into shorter, more manageable sequences that are faster to process. 146
A.4 HAN performance on Pubmed abstract classification based on dataset size . 151
B.1 Descriptions of the labels used in each classification task. Histology is omitted because there are over 500 labels; these are available in the SEER coding manual at https://seer.cancer.gov/tools/codingmanuals/ 154
B.2 Classifier f-scores for each task by class. F-scores are calculated using a one-vs-all setup for the class being examined. For histology, we only show the 20 most common classes in the test set. We note that not all possible classes appear in the test sets for each task. 159
B.3 McNemar's test p-values for site task. 164
B.4 McNemar's test p-values for laterality task. 164
B.5 McNemar's test p-values for behavior task. 164
B.6 McNemar's test p-values for histology task. 164
B.7 McNemar's test p-values for grade task. 165

B.8 Accuracy and macro F-score (with 95% confidence intervals) on each classification task using a reduced test set of 9481 pathology reports. Reports in this reduced set belong to a Tumor ID that is associated with only a single pathology report, thereby eliminating any addendums and supplementary reports. 165
B.9 Test set accuracy and macro F-score (with 95% confidence intervals) on each classification task for experiments on varying the number of attention heads used in the HiSAN. 165
B.10 The HiSAN architecture is similar to our previous work on HCANs; however, we found that certain components of the HCAN either decreased performance or provided no consistent improvement on performance when applied to cancer pathology reports – these components include using position embeddings, using a convolution window of size 3 instead of size 1 for Q, K, and V, using convolutions in the target attention, using layer normalization, and using two parallel self-attention layers. The following shows an ablation study of how different HCAN components affect test set accuracy and macro F-score (with 95% confidence intervals) on each classification task. 167
B.11 In our main set of experiments, we replace rare words (appearing fewer than five times in the corpus) with an "unknown_word" token. Below, we also show the performance of the HiSAN in which all possible words are included in the word embedding vocabulary matrix. Including all words gives small boost in performance in the histology task, which has the highest number of possible classes and highest class imbalance; however, it decreases performance on all other tasks. 167
C.1 Accuracy and Macro F-Score (with 95% confidence intervals) of modular and end-to-end training using the CNN baseline model. Results are grouped in pairs – for each technique for capturing report level context, we compare the modular results with the end-to-end results. 174

List of Figures

1.1 The architecture for our Hierarchical Attention Network. 13
1.2 The architecture for our Hierarchical Self-Attention Network. 15
1.3 The baseline case for classifying clinical reports and our methods for incorporating case-level context from other reports. In the figures above, Model represents an arbitrary deep learning model designed for text classification, the output of which is an embedding representation of the input document. . 17

2.1 Architecture for our Hierarchical Attention Network (HAN). The HAN first produces line embeddings by processing the word embeddings in each line. The HAN then produces a document embedding by processing all the line embeddings in the document. The final document embedding can then be used for classification or pre-training purposes. 30
2.2 HAN train and validation accuracies with and without pretraining during first 10 epochs for (A.) primary site classification task and (B.) histological grade classification task. 36

2.3 HAN annotations on sample pathology report for each classification task. The most important words in each line are highlighted in blue, with darker blues indicating higher importance. The most important lines in the report are highlighted in red, with darker red indicating higher importance. For each task, the HAN can successfully locate the specific line(s) within a document and text within that line that identify the primary site (e.g., lower lobe) or histological grade (e.g., poorly differentiated). The RNN structure utilized by the HAN allows it to take into account word and line context to better locate the correct text segments. 38
2.4 HAN document embeddings reduced to 2-dimensions via PCA for (A.) primary site train reports, (B.) histological grade train reports, (C.) primary site test reports, and (D.) histological grade test reports. 39
2.5 Confusion matrix for (A.) HAN w/ pretraining on primary site classification task and (B.) HAN w/o pretraining on histological grade classification task. 39

3.1 Architecture for our Hierarchical Self Attention Network (HiSAN). 54
3.2 Types of attention mechanisms used in the HiSAN. 56
3.3 Validation accuracy vs time on site classification task. 65
3.4 The HiSAN encodes each document into an fixed-size vector representation (aka "document embedding") that we can visualize in 2D using t-SNE. The left figure shows the document embeddings for all entries in the test set and right figure shows the document embeddings for all misclassified entries in the test set for the site task. 69
3.5 A section of a pathology report on breast cancer. In the top figure, we can follow the attention weights across all the attention layers in the HiSAN to visualize the words used to make the final classification decision. In the bottom figure, we can examine the self-attention weights associated with a particular word (in this case, the word "breast") to visualize how the HiSAN finds relationships between individual words. We note that for both figures, we show the aggregate attention weights across all attention heads in the HiSAN. 71

4.1 The baseline case for classifying EHRs and the five methods for incorporating case-level context from other reports. In the figures above, "Model" represents an arbitrary deep learning model designed for text classification, the output of which is an embedding representation of the input document. 80
4.2 The top two subfigures show the cancer site document embeddings generated by the HiSAN for each pathology report in our test set with and without the self-attention module for capturing case-level context. The bottom two figures only show the document embeddings of misclassified reports in our test set. All document embeddings are colored by the ground truth organ system and visualized using t-SNE. 94
4.3 The cancer site document embeddings generated by the HiSAN for the pathology reports associated with four unique tumor IDs, with and without the self-attention module for capturing case-level context. These figures share the same axes as Figure 4.2 and thus can be directly compared. Within each of the four trajectories, document embeddings are numbered from earliest to latest and are colored by the predicted organ system. We notice that without case-level context, reports belonging to the same tumor ID are classified under different organ systems. Adding case-level context addresses this problem and all document embeddings from the same tumor ID are placed in the same location in the embedding space. 97

5.1 The architecture for our Hierarchical Convolutional Attention Network. . . . 110
5.2 Validation accuracy vs time on Amazon sentiment analysis task. 112
A.1 Preprocessing pipeline for our cancer pathology report dataset 138
A.2 (A.) Average number of lines per report and (B.) average number of words per line across all 2505 pathology reports in corpus 139
A.3 TF-IDF weighted word embeddings reduced to 2-dimensions via PCA for (A.) primary site train reports, (B.) histological grade train reports, (C.) primary site test reports, and (D.) histological grade test reports 143

A.4 Doc2Vec embeddings reduced to 2-dimensions via PCA for (A.) primary site train reports, (B.) histological grade train reports, (C.) primary site test reports, and (D.) histological grade test reports 144
A.5 CNN document embeddings reduced to 2-dimensions via PCA for (A.) primary site train reports, (B.) histological grade train reports, (C.) primary site test reports, and (D.) histological grade test reports 145
A.6 Naive Bayes Confusion Matrix for (A.) Primary Site and (B.) Histological Grade 146
A.7 Logistic Regression Confusion Matrix for (A.) Primary Site and (B.) Histological Grade 147
A.8 Support Vector Machine Confusion Matrix for (A.) Primary Site and (B.) Histological Grade 147
A.9 Random Forest Machine Confusion Matrix for (A.) Primary Site and (B.) Histological Grade 148
A.10 Extreme Gradient Boosting Confusion Matrix for (A.) Primary Site and (B.) Histological Grade 148
A.11 Recurrent Neural Network without Attention Confusion Matrix for (A.) Primary Site and (B.) Histological Grade 149
A.12 Recurrent Neural Network with Attention Confusion Matrix for (A.) Primary Site and (B.) Histological Grade 149
A.13 Convolutional Neural Network Confusion Matrix for (A.) Primary Site and (B.) Histological Grade 150
A.14 The probability distributions of the 100 most common words in the Pubmed corpus with 8,000 abstracts, 80,000 abstracts, and 800,000 abstracts 152
B.1 Histograms of the number of occurrences per label for each of the five classification tasks, arranged from most common to least common. For the site and histology tasks, we only show the 50 most common labels. 155

B.2 Confusion matrices for the HiSAN's predictions on the test set for the different classification tasks (continued on following pages). Histology is excluded due to the large number (500+) of possible labels. For each task, we display the confusion matrix with and without normalization by class support size (number of elements in each class). 160
B.3 The attention weights assigned by different attention heads to each word in the same cancer pathology report for the site classification task. We see that different attention heads learn to pay attention to different sets of words. . . 166
C.1 Histogram of number of pathology reports associated with each unique tumor ID. 168
C.2 Histograms of the number of occurrences per label for each of the six classification tasks, arranged from most common to least common. For the site, subsite, and histology tasks, we only show the 50 most common labels. Detailed information about each label can be found online in the SEER coding manual at https://seer.cancer.gov/tools/codingmanuals/. (Figure continued on next page.) 169

Chapter 1

Introduction

Electronic health records (EHRs) are the prevalent method for documenting and storing clinical outcomes in the US. In 2017, the Office of the National Coordinator for Health Information Technology reported that 86% of office-based physicians store health records electronically[1]. Because EHRs are designed to capture information from all clinicians involved in the care of patients, they represent a rich source of clinical data – information may include demographics, progress notes, medications, vital signs, past medical history, immunizations, laboratory tests and results, radiology reports, and more[2]. As a result, EHRs can be a useful tool in monitoring trends in public health, tracking the spread of communicable diseases, and studying the effectiveness of different treatments.

Unfortunately, effectively extracting and utilizing the information within EHRs can be challenging, time-consuming, and expensive. There can be significant variation in how data is organized within EHRs [45, 129, 1, 22, 104], making it difficult to develop automated approaches that are consistent across different EHRs. In addition, human mistakes are common within the various phases of the lifetime of an EHR – in the diagnosis process prior to EHR creation [134, 103], in data entry during the creation process of the EHR [89, 6], and in interpretation of the EHR after it has been created [127, 12]. Furthermore, manual information extraction from EHRs requires expert knowledge and is extremely time consuming [95]. Developing effective automated methods for information extraction and

[1] https://dashboard.healthit.gov/quickstats

[2] https://www.healthit.gov/faq/what-information-does-electronic-health-record-ehr-contain

classification of EHRs and clinical records is an important step toward large scale utilization of the data contained within EHRs.

Critical information within EHRs is often presented in the form of free form text [70, 34]. Unlike structured EHR information, which is relatively easy to extract using database systems or rule-based methods [51], automated information extraction and classification of unstructured text requires natural language processing (NLP) tools; even the best of these are prone to errors due to to the inherent variability of human language [105]. In the domain of EHRs and clinical text, most state-of-the-art approaches utilize shallow, word-level convolutional neural networks (CNNs) [55, 18, 15, 102, 7, 49, 123], shallow recurrent neural networks (RNNs) [121, 85, 113, 146, 58], or traditional machine learning approaches such as Naive Bayes, logistic regression, and support vector machines [60, 53, 38, 67]. These methods typically date back to 2015 or earlier and significantly lag behind the state-of-the-art approaches used by the NLP community for other tasks such as sentiment analysis, machine translation, and question answering [156, 131]. Furthermore, rule-based approaches, which are seldom used in state-of-the-art approaches for general NLP tasks, are still commonly utilized in many recent clinical NLP approaches [144].

In this work, we focus on the specific application of cancer pathology reports due to (1) the importance of this problem area and (2) recent opportunities of data access. Cancer is the second leading cause of death in the United States [132]. To track aggregate cancer statistics and trends, the National Cancer Institute's (NCI) Surveillance, Epidemiology, and End Results (SEER) program collects and archives cancer pathology reports from cancer cases across the US [3]; these pathology reports contain detailed information including patient histories, treatments, and outcomes. Currently, trained human annotators must manually extract critical data elements, such as site, histology, behavior, and grade, from these pathology reports. In 2016, NCI and several SEER cancer registries partnered with several Department of Energy (DOE) national laboratories to develop automated information extraction and classification approaches for cancer pathology reports to ease the burden on human annotators, of which this work is a part. This partnership not only opened an unprecedented trove of clinical text data for the development of modern approaches towards

[3] https://seer.cancer.gov

clinical NLP, but also catalyzed the modernization of existing systems for tracking cancer trends and treatment outcomes in the US. The end goal is to develop and implement an accurate, effective, and timely platform for automatically extracting key data elements from the hundreds of thousands of pathology reports generated annually, which would greatly enhance existing cancer surveillance capabilities and help build the foundations for additional data-driven applications such as predicting patient trajectories and recommending specific treatments.

As part of the DOE-NCI partnership, this work seeks to bridge the gap between modern approaches for general NLP and the NLP approaches used on clinical text. We focus on deep learning approaches due to their proven effectiveness across a wide range of general NLP tasks, which is discussed further in Section 1.1. However, we note that architectures which perform well on general NLP tasks do not always translate well into clinical NLP due to notable differences between general text and clinical text; these differences include length, grammar, and signal-to-noise ratio and are discussed in greater detail in Section 1.1. Taking these into consideration, we focus on the following research questions:

1. Can hierarchical neural architectures, which have been shown to be highly effective in classifying general text such as news snippets and Yelp reviews, translate well into the clinical NLP domain and perform better than the existing state-of-the-art methods for classifying key data elements in cancer pathology reports?

2. What architectural improvements can be made to these hierarchical networks to adapt them to the nuances specific to clinical text and maximize their accuracy and speed?

Our work in this area has resulted in major improvements over the previous state-of-the-art in identifying key data elements in cancer pathology reports, a novel hierarchical architecture designed specifically for cancer pathology reports and other clinical text, and the development of methods for analyzing a series of clinical reports in context of one another that has been previously unexplored in the clinical NLP domain.

1.1 State of the Art

1.1.1 General NLP

NLP is a highly active area of research with many applications and subfields, such as question answering [57], machine translation [42], and sentiment analysis [97]. Although each subfield has its own specialized methods and techniques geared towards specific applications, there are general trends and techniques that span NLP as a whole.

In recent years, deep learning has become a dominant method for NLP due to its proven effectiveness in a wide range of tasks [156]. Prior to deep learning, NLP systems often relied on manual feature engineering, such as TF-IDF or hand-engineered rules. These had significant shortcomings, such as being tedious and time-consuming to build and not always scaling well to new text that is different from that which was used to develop the system [19].

Deep learning addresses this by learning salient features directly from the raw text itself, and in many cases it removes the need for manual feature engineering. To achieve this, deep learning approaches generally utilize word embeddings as input [66], which are learned vector representations of individual words. Not only can word embeddings capture complex syntactic, semantic, and linguistic properties of words, but newer approaches are also capable of capturing the meaning of new words outside the training corpus [11, 160], enabling deep learning approaches to better adapt to new situations. Combined with the ability of neural architectures for capturing complex non-linear patterns, word embeddings enable deep learning to be highly effective in complex NLP applications where manual creation of rules or features is infeasible.

In deep learning applications for NLP, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are two highly-popular families of architectures [156]. Originally designed for image processing, CNNs utilize a convolution filter that slides across the text, examining n consecutive words at a time [68]. CNNs are fast and easy to parallelize; however, because they utilize a sliding window that can only examine n words at a time, they have difficulty capturing long-range linguistic dependencies that exceed n words. To deal

with this, newer architectures stack multiple CNN layers that extend the network's ability to capture patterns across longer text segments [25, 64, 33].

RNNs are a neural architecture designed for sequential inputs [83]. In NLP, RNNs take in a sequence of words; when processing each word, the RNN accounts for information from all previous words in the sequence [136]. Therefore, RNNs are capable of learning linguistic patterns and dependencies over arbitrary lengths of text. Unfortunately, the computational complexity of RNNs increases noticeably for longer input sequences, and the backpropogation operations are difficult to parallelize [76, 87]. As a result, RNNs are much slower to train than their CNN counterparts.

Self-attention, a relatively new neural architecture that gained popularity in 2017, attempts to address the shortcomings of both CNNs and RNNs [39, 138]. Self-attention takes a full sequence of words as input. It then generates a similarity matrix that compares each word in the sequence to all other words to find meaningful relationships. Finally, it outputs a new transformed sequence in which each transformed word has captured within it information from all other relevant words in the same sequence. Like RNNs, self-attention can capture long-range linguistic patterns, and like CNNs, self-attention is relatively fast to train and easy to parallelize [141, 77].

Depending on the NLP task, CNN, RNN, and/or self-attention modules often make up the base components of state-of-the-art systems. The general trend in NLP over the past few years has been toward deeper and more complex neural architectures, which in turn capture more complex and nuanced linguistic patterns relevant for a given NLP task. This includes intricate reasoning modules [143, 157, 149], character-level analysis [32, 81], and contextual word embeddings such as ELMo [112] and BERT [28]. The recent GPT-2 language model developed by Open AI utilizes 48 layers and over 1.5 billion learnable parameters [119]. As computation resources become more powerful and less expensive and textual training data becomes more widely available, it seems likely that the trend towards bigger and deeper neural networks will continue.

1.1.2 Clinical NLP

NLP has many important applications in the clinical domain, including missing data imputation [16], patient phenotyping [108], cohort identification [44], clinical decision support [126], and many more. While deep learning is commonly used in these clinical NLP tasks [131], clinical NLP deep learning architectures are often far less complex than those used in general NLP. Furthermore, in many clinical NLP applications, it is still common to see traditional machine learning approaches such as logistic regression and support vector machines [60, 53, 38, 67] and rule-based systems such as cTAKES [122, 153]. These traditional approaches often rely on TF-IDF features or manually engineered features which are unable to capture complex or nuanced linguistic patterns that may be important for recognizing important events or information in clinical text.

Two deep learning architectures that are widely used in clinical NLP are the word-level shallow and wide CNNs [68], first developed in 2015, and shallow RNNs utilizing gated recurrent units [21] or long short-term memory units [50], which were respectively developed in 2014 and 1997. Despite the relative simplicity of these architectures, they comprise the current state-of-the-art across various clinical NLP applications [96, 49, 121, 85, 113], and many recent advancements in clinical NLP are minor variations of these two architectures [163, 47, 124].

There may be several explanations for why the state-of-the-art in clinical NLP lags behind the state-of-the-art in general NLP when it comes to model complexity. First, clinical text is often much longer than text used for many general NLP tasks, such as those in the General Language Understanding Evaluation [142] and Natural Language Decathalon [91] benchmarks. Complex deep learning architectures developed for many general NLP typically utilize input text no longer than a single paragraph in length; these architectures may be too computationally expensive to use on clinical notes that are often several pages long.

Second, clinical text is often grammatically inconsistent compared to the text used in general NLP tasks, such as Wikipedia articles for question answering or book passages for machine translation. A clinical text document may contain a combination of notes written by a clinician, semi-structured lists of patient history and test outcomes, and tabular structures

containing detailed information. This added complexity presents additional challenges for deep learning architectures, which must learn how to effectively parse different types of input.

Third, compared to general NLP tasks such as sentiment analysis or machine translation, the signal to noise ratio is often far lower in clinical NLP tasks. For example, a multi-page EHR may contain detailed information about patient demographics, patient history, procedures, and/or drug prescriptions, but only a few words may be relevant for identifying the presence of a particular disease.

Fourth, establishing gold-standard labels for supervised learning on EHRs often requires expensive expert knowledge, so labelled data is much harder to come by for applications in the clinical domain. Consequently, the available labelled datasets tend to be small in size (often fewer than 2000 labelled samples), thereby reducing the effectiveness of data-hungry and complex deep learning architectures with millions of learnable parameters.

Finally, access to clinical data is often restricted due to regulations protecting patient privacy; therefore, compared to other open access datasets for general NLP tasks, NLP researchers may have more difficulty accessing data for the development of more modern and advanced clinical NLP approaches. Because of this, clinicians with access to the data and who lack specialized NLP expertise may be limited to older and more simple approaches that are easiest to implement.

The basic CNN and RNN architectures commonly used in clinical NLP have several notable shortcomings that may reduce their effectiveness in clinical NLP tasks – namely, CNNs may have difficulty to capturing long-range linguistic relationships useful for NLP tasks and RNNs can be prohibitively slow to train on long clinical texts. This work introduces architectural improvements that address these issues and help bridge the gap between the state-of-the-art in clinical NLP and general NLP.

1.1.3 Classifying Cancer Pathology Reports

In 2019, approximately 1.7 million new cases of cancer are expected to be diagnosed and approximately 600,000 cancer deaths are expected to occur in the US alone[4]. Based off existing trends, each year 440 per 100,000 people are diagnosed with cancer and 160 per

[4]https://www.cancer.org/research/cancer-facts-statistics

100,000 people die from cancer; approximately 38.4% of people will be diagnosed with cancer at some point in their lifetimes.

When a clinician suspects that a patient has cancer, a tissue sample is sent to a pathologist; the pathologist then performs a detailed diagnosis of the tissue sample to determine the existence and extent of the cancer and documents all findings within a cancer pathology report [26]. Each report typically includes a wide range of information including patient information such name and birth date, a gross description of the tissue as seen by the naked eye, a description of the cells seen under a microscope, a diagnosis of the tumor and grade (how abnormal the cells look and how quickly the tumor is likely to grow and spread), the tumor size and margins, notes about additional tests or opinions, and information about the pathologist and laboratory. Because the cancer pathology report is the most comprehensive diagnosis for a potential cancer case, it plays the definitive role in cancer diagnosis and is essential in the selection of an optimal treatment plan.

Cancer registries are organizations that collect, manage, and analyze cancer pathology reports in order to track aggregate cancer trends and survival rates, guide cancer prevention and control programs, evaluate the effectiveness of cancer treatments, and support cancer research. NCI's SEER program works with cancer registries across the US; collectively, SEER registries cover 34.6% of the US population[5]. Due to its high coverage, the SEER program is an authoritative source for cancer surveillance information in the US.

In order to track trends effectively, the SEER program collects detailed data from each pathology report including patient demographics, primary tumor site, tumor morphology and stage, first course of treatment, and follow-up status5[0]. Currently, trained experts at the SEER registrars are tasked with extracting this information manually – this is a labor-intensive process that requires the human expert to read each pathology report and annotate the key data elements that need to be tracked. Given the high volume of cancer pathology reports generated each year and the limited resources of cancer registries, there is an urgent need to develop accurate and effective automated methods that can assist or replace humans in classifying and extracting information from these reports.

[5]https://seer.cancer.gov/about/overview.html

Most existing methods for automated information extraction and classification of cancer pathology reports rely on rule-based approaches [14, 100, 145, 73, 148, 101, 24] or traditional machine learning approaches based of n-gram features, such as Naive Bayes, logistic regression, Adaboost, support vector machines, and conditional random fields [88, 80, 150, 154, 162, 144]. As mentioned previously, these approaches have significant weaknesses – hand-engineered rules can be brittle and fail when a document does not adhere to the pre-determined rules, while n-gram features are unable to account for any potentially important linguistic patterns that span beyond a short window of text [114, 160]. Furthermore, the datasets used for previous work are often limited in size and scope – these usually have fewer than 2000 labelled samples and may only include certain types of cancers. Therefore, it is uncertain if approaches developed on these datasets can effectively be scaled toward widespread use in cancer registries that must identify all types of cancers.

Prior to this work, the state-of-the-art in classifying cancer pathology reports was a word-level shallow and wide CNN [116]. This network utilizes three parallel convolution layers that respectively examine a window of three, four, and five consecutive words at a time, followed by a maxpool over time to find the most salient combinations of three-, four-, and five-grams that occur in a document and are relevant to the given task. Consequently, this CNN suffers from the same weakness as n-gram based machine learning classifiers – it can only identify meaningful linguistic patterns no longer than five words in length. Furthermore, this CNN was developed on a relatively small dataset of less than 1000 labelled pathology reports covering only breast and lung cancer.

In this work, we push forward the state-of-the-art in cancer pathology report classification not only by introducing more modern deep learning approaches that can address the weaknesses of rule-based, machine learning, and simple CNN and RNN approaches, but also by evaluating our methods against very large corpora of hundreds of thousands of pathology reports covering all types of cancers occurring in the population.

1.2 Research Objectives and Approach

The purpose of this work is to show that hierarchical neural architectures can better capture the complex linguistic patterns necessary for classification of cancer pathology reports and other clinical texts relative to the basic CNN, RNN, and traditional machine learning techniques commonly used for clinical NLP. Our results to show meaningful improvements in classification accuracy across key data element classification tasks on cancer pathology reports; due to the importance of this problem area, even small improvements in accuracy and scope can impact a large number of potential patients. Furthermore, because clinical text is often much longer than the single sentences or short paragraphs used for general NLP tasks, we focus on approaches that are not prohibitively expensive in terms of time and/or memory usage, which is essential to make our methods practical for users and organizations who may have limited computational resources and/or large collections of training data.

Our novel contributions are as follows:

1. We test the effectiveness of using a hierarchical neural structure that first breaks down long documents into smaller semantic chunks such as lines or sentences, then processes each smaller chunk individually before processing the entire document as a whole. For longer documents such as clinical reports, this hierarchical structure improves a neural architecture's ability to locate and retain relevant information for a given task. We incorporate this hierarchical structure into an RNN with attention and show this approach outperforms non-hierarchical CNNs and RNNs and traditional machine learning approaches for information extraction from long pathology reports.

2. Because RNNs are inherently slow to train and difficult to parallelize, we explore the effectiveness of replacing RNNs with self-attention layers while still maintaining a hierarchical structure. We show that compared to RNNs, self-attention can speed up training time on a large corpus of cancer pathology reports by an order of magnitude while also giving a small boost in classification performance and aiding in interpretability. Our results suggest that a hierarchical self-attention network can provide better accuracy than a basic CNN without being prohibitively slow to train.

3. Clinical reports often come in a sequence – for example, a single patient may generate multiple related reports over the trajectory of a disease or a cancer case. We therefore extend our hierarchical structure one additional level such that rather than looking only at individual reports, it concurrently examines all reports within a sequence so that case-level context may be incorporated during the information extraction process for each report. We show that incorporating case-level context gives a significant boost in the performance of various classification tasks over cancer pathology reports.

Our datasets consist of cancer pathology reports obtained from various SEER tumor registries. Given a set of cancer pathology reports, our task is to extract key data elements from each report, such as the site, subsite, and histology of the cancer described in the report. Our ground truth labels are based off annotations from human experts. Because we obtained data access to additional tumor registries over the course of our research, the exact size of the datasets used in our experiments varies across each individual study. In each experiment, we compare our methods against a wide variety of traditional machine learning classifiers commonly used for clinical NLP as well as the previous state-of-the-art in classifying cancer pathology reports, the word-level shallow and wide CNN [116].

Our pathology report datasets reflect the distribution of cancer cases collected by the cancer registries and therefore are subject to extreme class imbalance – for example, the number of samples for the rarer cancer types may be extremely low compared to the more prevalent cancer types. To account for this, all experiments are evaluated using two distinct metrics – overall accuracy, which measures overall performance across all classes, and macro F-score, which equally weighs the performance on each class and thus better reflects performance on the rare classes. We note that in classification tasks such as ours in which each report is assigned to exactly one class, accuracy is the same as micro F-score.

1.2.1 Hierarchical Attention Networks

Hierarchical attention networks (HANs) were first developed in 2016 by Yang et. al. for classification of short paragraphs [152]. At the time of its release, HANs achieved new state-of-the-art results across a wide range of text classification and sentiment analysis tasks, including on Yelp reviews, Amazon reviews, and Yahoo Answers.

General RNN-based architectures have difficulty retaining important information over very long text segments, even with the addition of an attention mechanism [139]. The HAN addresses this problem by first breaking up a long text into shorter segments, such as individual sentences. The HAN utilizes a hierarchical structure in which the lower hierarchy creates a latent representation of each shorter text segment, and then the upper hierarchy analyzes the entire text using the representations created from the lower hierarchy. Because clinical reports can often exceed several pages in length, the HAN performs better than non-hierarchical approaches such as the basic CNNs and RNNs that are commonly used for clinical NLP.

In our first set of experiments, we evaluate the performance of the HAN against other deep learning and traditional machine learning techniques in classifying cancer site and histological grade from a corpus of approximately 1000 cancer pathology reports. The architecture for our HAN is shown in Figure 1.1. Similar to the original HAN paper, we utilize two hierarchies, each one composed of an RNN with an attention mechanism. Unlike restaurant and product reviews, which can usually be broken down into sentences, cancer pathology reports are often written using both paragraphs of grammatically correct text as well as lines which list relevant diagnostic information. Therefore, we break down each pathology report into smaller text segments using both the naturally occurring linebreaks and sentence punctuation in the report. The lower hierarchy generates a latent representation of each line/sentence which captures the relevant information within that line/sentence, and the upper hierarchy generates a latent representation of the entire report based off the most important lines/sentences generated by the lower hierarchy. This final document representation is then used for classification.

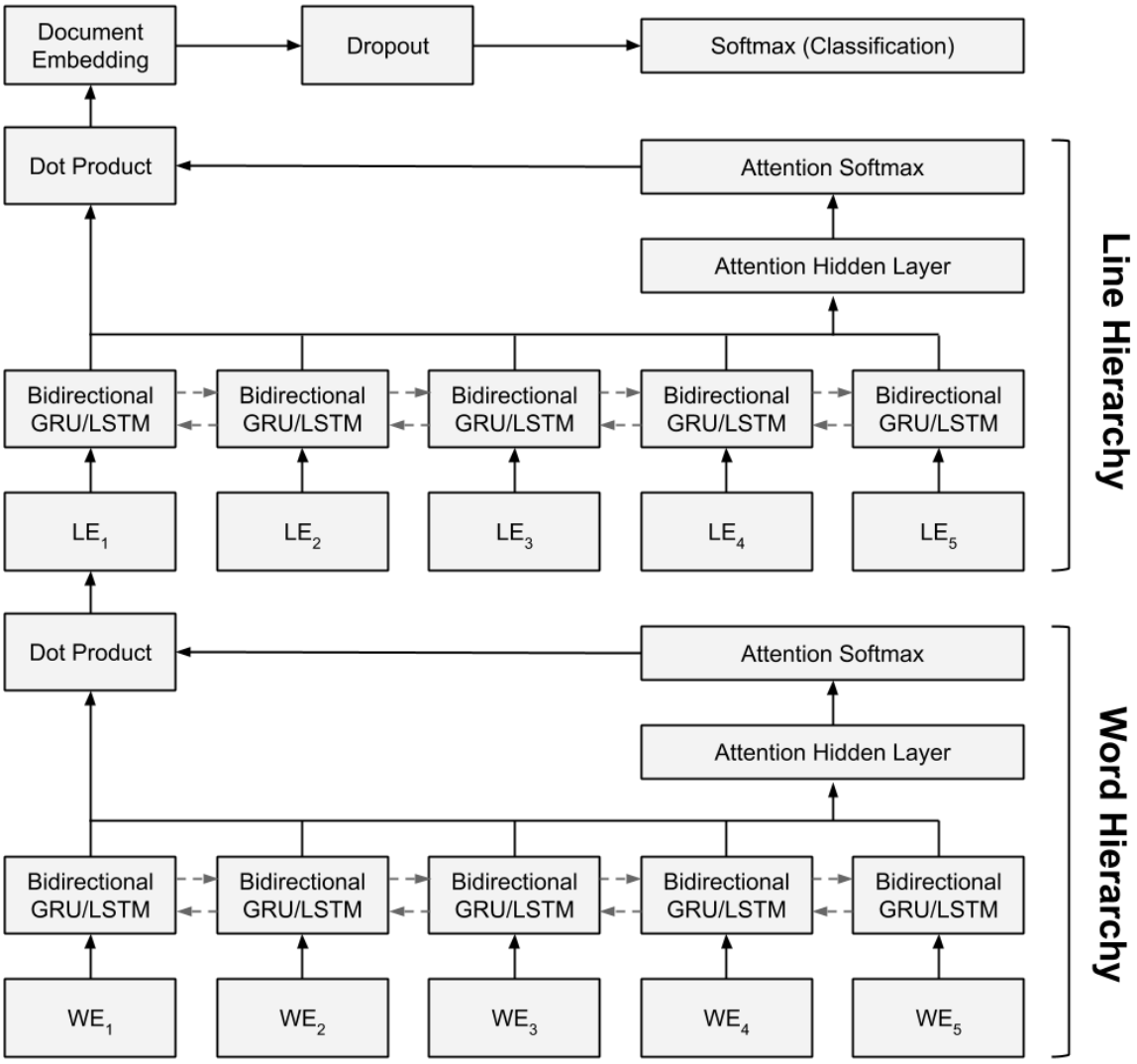


Figure 1.1: The architecture for our Hierarchical Attention Network.

Because the HAN utilizes an RNN-based architecture in combination with a hierarchical structure, it can identify linguistic patterns spanning much longer text spans than CNNs or n-gram-based traditional machine learning classifiers, which can only identify patterns within a small pre-defined window size. Our results (discussed in Chapter 2) show that the HAN achieves significantly better accuracy than CNNs and traditional machine learning approaches in identifying site and histological grade in cancer pathology reports.

1.2.2 Hierarchical Self-Attention Networks

One weakness of HANs is that it relies on an RNN-based architecture, which can be prohibitively slow to train on large datasets. Our experiments (discussed in Chapter 3) show that HANs can be 10x-50x slower than CNNs depending on the lengths of the input documents. This can be a significant challenge for model development and hyperparameter tuning in clinical applications that involve large corpora of thousands or millions of clinical documents.

In 2017, Vaswani et. al. demonstrated that replacing RNN layers with self-attention layers results in a significant speed boost without sacrificing accuracy in machine translation [141]. We therefore develop a novel architecture similar to the HAN in which the RNN layers are replaced with self-attention; we name our architecture the hierarchical self-attention network (HiSAN). The architecture of our HiSAN is shown in Figure 1.2.

Because the HiSAN also utilizes a hierarchical architecture like the HAN, the HiSANs achieves similar performance to the HAN but trains much faster. We evaluate the performance of the HiSAN against the HAN and other deep learning and traditional machine learning classifiers in classifying cancer pathology reports across five key data elements – site, subsite, laterality, behavior, and grade – using a corpus of approximately 375,000 cancer pathology reports. Our results (discussed in Chapter 3) show that HiSANs train up to 10x faster than the HAN without any loss in accuracy. This makes the HiSAN a more accessible model for facilities with low computational resources; additionally, it is more practical for situations that require training on large datasets or retraining regularly on new data.

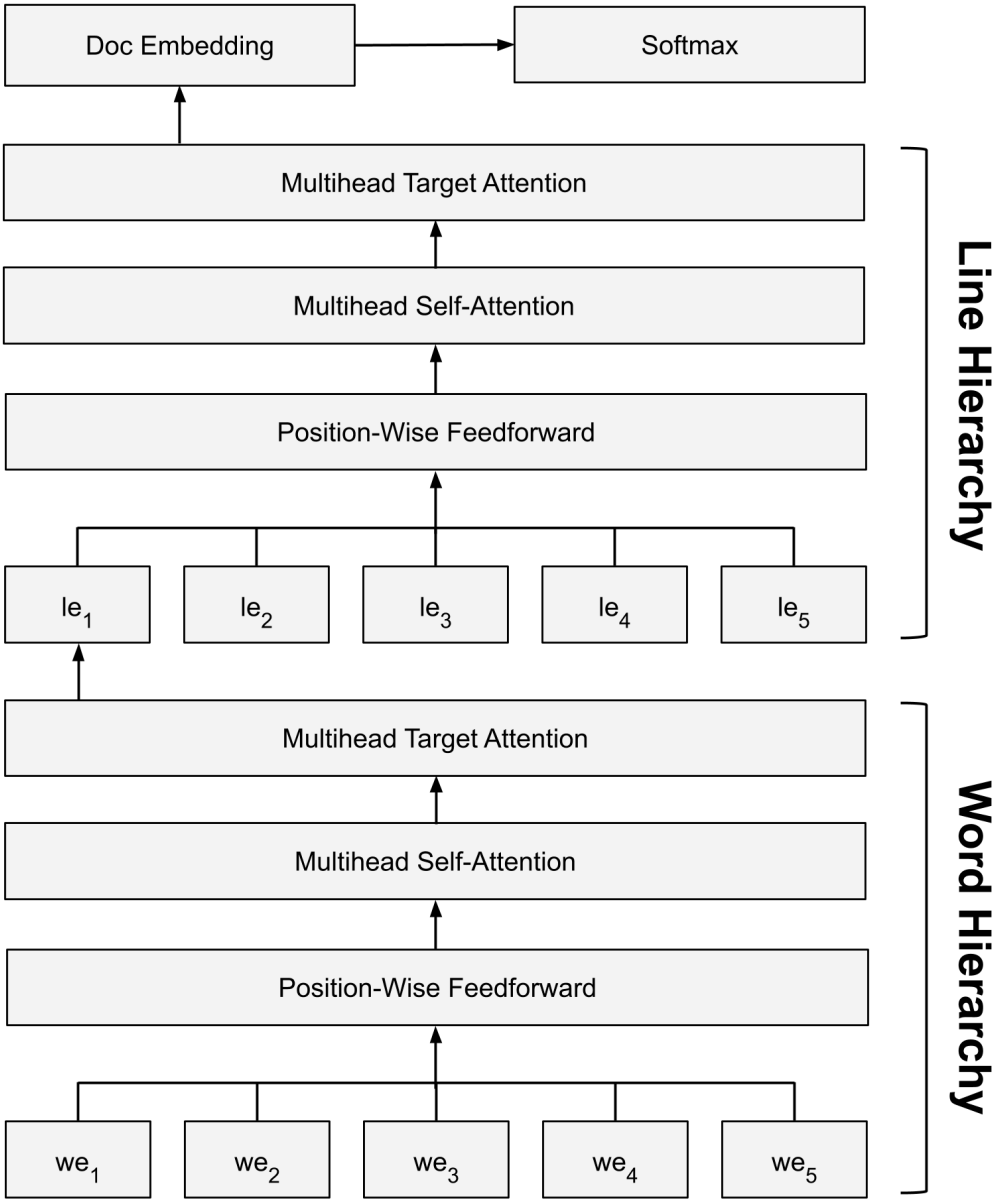


Figure 1.2: The architecture for our Hierarchical Self-Attention Network.

1.2.3 Capturing Case-Level Context

Clinical reports such as cancer pathology reports often come in a sequence – for example, a single patient may generate multiple reports over the trajectory of a disease or cancer case, and the information within each report is often related to the information in previous and future reports. Utilizing contextual information from other reports in the same sequence has the potential to yield major gains in information extraction and classification tasks - for example, certain ICD codes may be more likely to occur if a certain disease was previously detected or if a certain test was previously performed. Unfortunately, capturing and utilizing this type of case-level context has so far been unexplored in the clinical domain.

Existing approaches for information extraction and classification of clinical reports exclusively focus on individual reports only without accounting for contextual information from other reports. We therefore extend the hierarchical structure used in the HAN and HiSAN to also capture document-level relationships between clinical reports from the same related sequence (e.g., cancer pathology reports belonging to the same unique tumor). Using this framework, we explore several different methods for capturing case-level context, which are shown in Figure 1.3 – these include concatenation, RNNs, RNNs with linear-chain conditional random field (CRF), self-attention, and self-attention with linear-chain CRF.

We test our methods using a corpus of approximately 430K cancer pathology reports that are each labelled with six key data elements – site, subsite, laterality, behavior, histology, and grade. Each pathology report is also labelled with a unique tumor ID; for any given pathology report, we can therefore also incorporate contextual information from other reports belonging to the same tumor ID. Our methods that incorporate case-level context significantly outperform the baseline in which no case-level context is used.

While our methods for capturing case-level context are inspired by the hierarchical structure used in the HAN and HiSAN, we design them as modular add-ons such that the component for capturing case-level context can be trained independently on top of a base model for classifying individual reports without report level context. As a result, our approaches are compatible with most existing deep learning text classification models designed to classify individual documents and require minimal additional engineering to use.

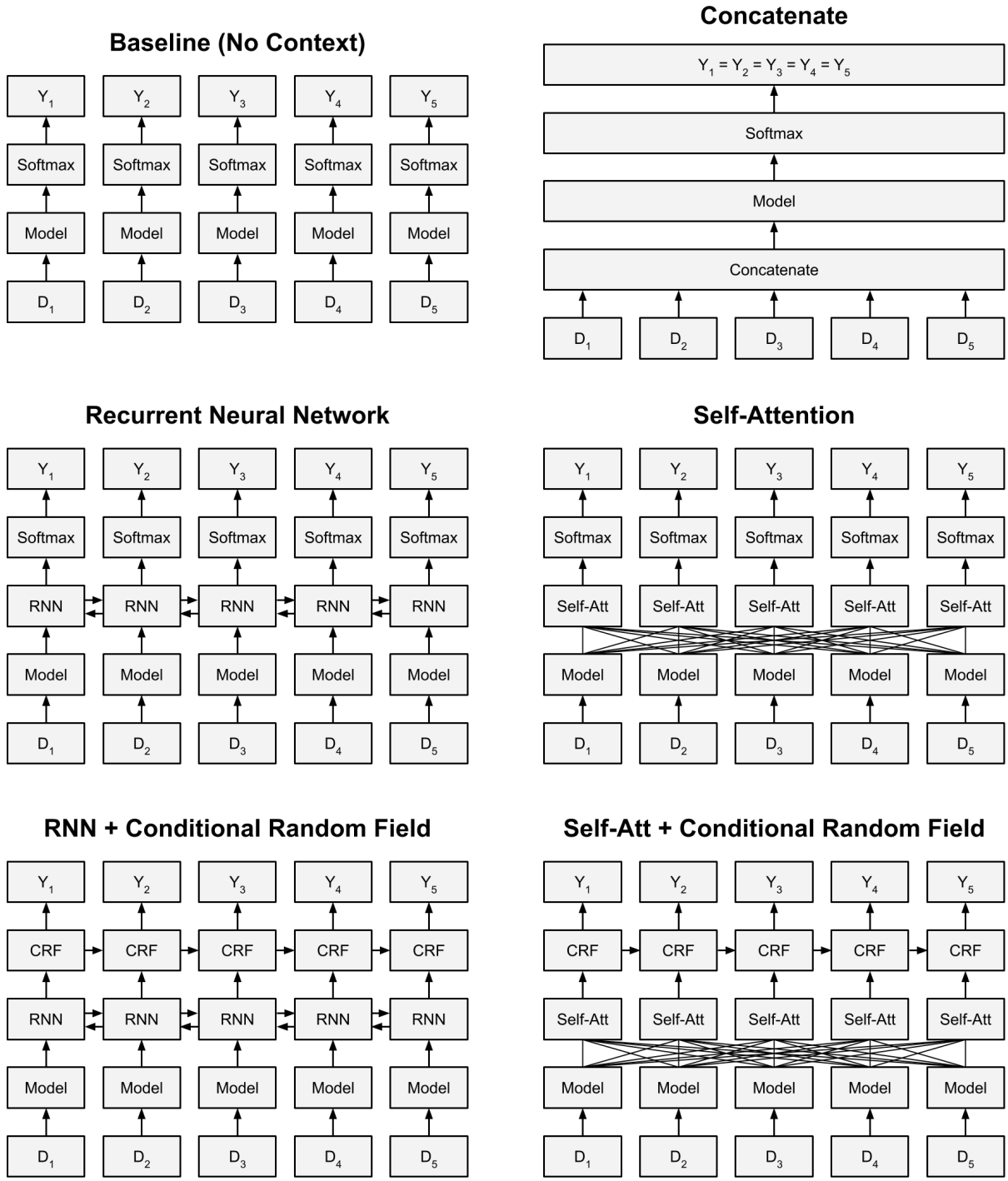


Figure 1.3: The baseline case for classifying clinical reports and our methods for incorporating case-level context from other reports. In the figures above, Model represents an arbitrary deep learning model designed for text classification, the output of which is an embedding representation of the input document.

Therefore, we expect that with minimal modifications, our modular add-ons may be applied across a wide range of EHR- and clinical text-based tasks beyond cancer pathology reports.

1.3 Dissertation Outline

This dissertation is a manuscript-style dissertation in which each of the three core chapters is a previously published journal paper or is a paper currently undergoing the submission process.

Chapter 2 is a version of our paper "Hierarchical Attention Networks for Information Extraction from Cancer Pathology Reports," which was originally published in the Journal of the American Medical Informatics Association. This chapter introduces the Hierarchical Attention Network architecture, which was originally developed by Yang et. al. in 2016 [152] and shown to be effective for a wide range of general text classification tasks. We apply this network on a corpus of approximately 1000 cancer pathology reports and show that it works better than previous methods (including a basic CNN, a basic RNN, and traditional machine learning methods) on two classification tasks – primary site classification and histological grade classification.

Chapter 3 is a version of our paper "Classifying Cancer Pathology Reports with Hierarchical Self-Attention Networks," which was recently accepted for publication in the journal Artificial Intelligence in Medicine. This chapter introduces the Hierarchical Self-Attention Network, which is a novel architecture that overcomes some of the weaknesses of the HAN associated with slow training time. We test the performance of the HiSAN on a corpus of approximately 375K cancer pathology reports using five different classification tasks – site, laterality, behavior, histology, and grade. We show that the HiSAN not only achieves better accuracy and macro F-score than the HAN and other classification methods, but also trains an order of magnitude faster than the HAN.

Chapter 4 is a version of our paper "Using Case-Level Context to Classify Cancer Pathology Reports," which is currently in the submission process to an academic journal. This chapter explores how to use hierarchical methods to incorporate case-level context from a sequence of related cancer pathology reports, which has previously been unexplored in the

clinical NLP domain. We compare several hierarchical methods for capturing case-level context to the baselines of using no case-level context (i.e. using a text classification method such as a CNN or HiSAN on a single document) or simply concatenating all reports in the sequence into a single long document before feeding it into a text classification model. We test our methods on a corpus of approximately 550K cancer pathology reports using six different classification tasks – site, subsite, laterality, behavior, histology, and grade. We show that our hierarchical methods for capturing case-level context give a significant accuracy boost compared to the baselines.

Finally, chapter 5 concludes this work by revisiting the main contributions and findings of our three core chapters in context of current developments and applications in the clinical and general NLP communities. We discuss additional experiments related to transfer learning, how our hierarchical architectures compare to the latest work from the general NLP community, the implications and limitations of our work for the clinical NLP domain, and future work to extend the approaches developed in this thesis.

Chapter 2

Hierarchical Attention Networks for Information Extraction from Cancer Pathology Reports

A version of this chapter was originally published in the Journal of the American Medical Informatics Association:

Shang Gao, Michael T. Young, John X. Qiu, Hong-Jun Yoon, Blair Christian, Paul A. Fearn, Georgia Tourassi, and Arvind Ramanathan. "Hierarchical attention networks for information extraction from cancer pathology reports." Journal of the American Medical Informatics Association, 25(3):321330.

Shang Gao and Arvind Ramanathan conceptualized the study and developed the experimental setup. Shang Gao ran all experiments and wrote the manuscript. Michael T. Young, John X. Qiu, Hong-Jun Yoon, Blair Christian, and Paul A. Fearn supported the data collection process and help develop the data preprocessing pipeline. Arvind Ramanathan and Georgia Tourassi proofread and edited the final manuscript.

No revisions to this chapter have been made since the original publication.

2.1 Abstract

Objective: We explored how a deep learning approach based on hierarchical attention networks (HAN) can improve model performance for multiple information extraction tasks from unstructured cancer pathology reports compared to conventional methods that do not sufficiently capture syntactic and semantic contexts from free text documents.

Materials and Methods: Data for our analyses was obtained from 942 de-identified pathology reports collected by the National Cancer Institute (NCI) Surveillance, Epidemiology, and End Results (SEER) program. The HAN was implemented for two information extraction tasks, namely (1) primary site – matched to 12 ICD-O-3 topography codes (7 breast; 5 lung primary sites) and (2) histological grade classification – matched to G1-G4. Model performance metrics were compared to conventional machine learning (ML) approaches including Naive Bayes (NB), logistic regression (LR), support vector machine (SVM), random forest (RF), and extreme gradient boosting (XGB), and other deep learning (DL) models including a recurrent neural network (RNN), a recurrent neural network with attention (RNN w/A), and a convolutional neural network (CNN).

Results: Our results demonstrate that for both information tasks, HAN performs

significantly better compared to the conventional ML and DL techniques. In particular, across the two tasks, the mean micro and macro F-scores for the HAN with pretraining were [0.852,0.708], compared to NB [0.518, 0.213], LR [0.682, 0.453], SVM [0.634, 0.434], RF [0.698, 0.508], XGB [0.696, 0.522], RNN [0.505, 0.301], RNN w/A [0.637, 0.471], and CNN [0.714, 0.460].

Conclusions: HAN-based deep learning models show promise in information abstraction tasks within unstructured clinical pathology reports.

2.2 Objective

Cancer is the second leading cause of death in the United States [86]. Given the public health burden that cancer imposes on society, developing effective clinical surveillance of cancer remains one of the top priorities for the National Cancer Institute (NCI). The NCI Surveillance, Epidemiology, and End Results (SEER) program maintains a comprehensive database of diagnostic, treatment, and outcomes information from vast amounts of unstructured data sources such as clinical visit notes, pathology reports, treatment summaries, and so on. However, a critical challenge for leveraging these documents in the context of cancer surveillance is the abstraction of essential information regarding tumor type, location, and histological grade for a given patient at a given time.

The primary objective of our study is the design, implementation, and validation of a novel deep learning approach based on hierarchical attention networks (HANs) to automate information extraction from unstructured clinical pathology reports. We demonstrate that HANs can effectively capture the primary information regarding a patient's tumor type, grade, and location from unstructured text. Compared to traditional natural language processing (NLP) approaches, we show that HANs can enhance the efficiency and quality of data extraction across two distinct information extraction tasks, namely, identifying the primary site and histological grade of a tumor from these pathology reports. Taken together, our approach opens new opportunities for scalable and automatic information extraction from unstructured clinical pathology reports.

2.3 Background

A key goal within the NCI SEER program is to provide a comprehensive measurement of cancer incidence, morbidity, survival, and mortality for persons with cancer [99]. For this purpose, the SEER program relies on the acquisition of diagnostic, treatment, and outcomes information through manual information abstraction and information processing by expert staff from vast amounts of unstructured sources (e.g., clinical visit notes, pathology reports, treatment summaries, etc.). This manual processing of information imposes inherent limitations on the volume and types of information that can be collected. Furthermore, the reports can be highly variable because they are sourced from hundreds of different healthcare providers and across multiple laboratories. Additional variability is caused by human factors such as different interpretations of coding rules and human error. Thus, manual classification of cancer pathology reports can be a challenging and time consuming task that requires extensive training. With the continued growth in the number of cancer pathology reports, the rise in patient survival rates, and the increase in treatment complexity, cancer registries face a significant challenge in manually reviewing large volumes of reports.

One solution to this problem currently being explored is the application of artificial intelligence and machine learning to automatically read and extract information from cancer pathology reports. If effective, such a solution would significantly reduce delays in report processing and allow trained personnel to focus their time on analysis and research. However, the content in cancer pathology reports can be difficult for machines to process due to its high variability–these include misspellings and missing punctuation, clinical diagnoses interspersed with complex explanations, different terminology to label the same cancer, and information about multiple cancers included in a single report. Developing an automated solution with high accuracy and consistency across a wide selection of reports is therefore challenging.

Whereas traditional machine learning models require human experts to encode domain knowledge through feature engineering, deep learning approaches are able to learn salient feature representations through backpropagation on the given dataset. This makes deep learning especially apt for natural language processing (NLP) tasks where manual encoding

of features is both inefficient and impractical. Recent experiments have shown that deep learning approaches that generate their own features can achieve state-of-the-art results in NLP tasks such as question answering, part-of-speech tagging, and sentiment analysis [71]. Convolutional neural networks (CNNs), traditionally used for computer vision, have also been adapted for NLP tasks [68]. In computer vision, CNNs use a sliding window of learned filters to identify the important features in an image. In NLP, the filters of a CNN are instead trained to process segments of words to identify the word combinations that are most pertinent to a particular task.

Recurrent neural networks (RNNs) have been shown to be especially effective in NLP tasks thanks to their specialized architecture for processing time-series data [82]. Unlike traditional feedforward neural networks, which take in an input and produce an output, RNNs are designed to take in a series of inputs over time. The output of a recurrent neural network depends on not only the input at the current timestep, but also inputs in previous (and possibly future) timesteps. In NLP tasks, RNNs process one word at a time and then learn linguistic patterns based on different sequences of words.

In basic RNNs, input data is written into the RNN at every timestep, even if it is irrelevant to the task at hand. This results in a dilution effect over time, so basic RNNs are unable to retain information and find relationships over a large number of timesteps. In NLP, this is problematic when semantic meaning is distributed over a long sequence of words. Long short-term memory (LSTM) cells [50] and gated recurrent units (GRU)[21] are RNN architectures that address this problem by using gating functions that control the flow of information into and out of the RNN. These architectures selectively process and output information based on its relevance–inputs that constitute noise can be ignored, and important inputs can be stored and saved within the RNN until they are required (for details, see Appendix A.4). In NLP, LSTM- and GRU-based architectures are often used in question answering tasks in which semantic meaning is spread across multiple sentences [71].

Attention mechanisms can significantly boost the performance of RNNs by allowing them to further focus on timesteps that are most critical to the task at hand [46]. In regular RNNs, decisions are made simply by using the RNN output at the final timestep; this means the RNN must capture the essence of the entire input sequence in a single output vector, which

is not always effective. With attention, the RNN saves an output at every timestep, and the attention mechanism then selects and combines the most important outputs based on their relevance to the task. This gives the RNN far more expressive power by allowing the RNN to save useful information at every timestep and then later choose which outputs to use for decision making.

Yang and co-workers [152] developed a hierarchical attention network (HAN) for classification of Yelp and movie reviews that outperformed both RNNs and CNNs. HANs expand on RNNs by utilizing hierarchies of RNNs that process long texts by breaking them down into smaller, more manageable segments. In this paper, we show that a similar architecture can be applied to effectively classify cancer pathology reports in different classification tasks.

Approaches for automated document classification in biomedical informatics range from carefully crafted rule-based systems [14] to traditional machine learning classifiers based on hand-engineered features [88] to deep learning approaches that require no manual feature engineering [79]. Jouhet and colleagues tested the effectiveness of Naive Bayes and Support Vector Machines using TF-IDF features on classifying the ICD-O-3 topographical codes and the morphological axes of the ICD-O-3 code for cancer pathology reports [65]. Jagannatha and Yu showed that recurrent neural networks using LSTM or GRU units can effectively classify medical events from unstructured text in Electronic Health Record notes [58]. Qiu and co-authors demonstrated that convolutional neural networks can classify the ICD-O3 topographical codes with higher accuracy than Naive Bayes, SVMs, and other traditional machine learning classifiers [117].

In this paper, we use two cancer pathology report classification tasks to demonstrate HANs outperform not only traditional machine learning approaches, but also other deep learning architectures in the two classification tasks.

2.4 Materials and Methods

2.4.1 Dataset Description and Preprocessing

The SEER pathology reports are obtained from five cancer registries (CT, HI, KY, NM, Seattle) with the proper IRB-approved protocol. The full corpus consists of 2505 de-identified reports. Some, but not all of these pathology reports are manually annotated by cancer registry experts based on standard guidelines and coding instructions used in cancer surveillance, depending on the classification task of interest (see below). These annotations serve as the labels used for our information extraction tasks.

We focus on two information extraction tasks: (1) primary site and (2) histological grade classification. For the primary site classification task, we use a corpus of 942 de-identified pathology reports matched to 12 ICD-O-3 topography codes corresponding to 7 breast and 5 lung primary sites. For the histological grade classification task, we use a total of 645 labeled reports matched to four histological grades G1-G4. For both tasks, we limit our training corpus to pathology reports with a single topography code and histological grade sourced only from the "Final Diagnosis" section of the report. In Table 2.1, we describe the corpus for the individual tasks.

We follow a standard preprocessing pipeline for each document of our corpus, illustrated in Appendix A.1. The cancer pathology reports in our dataset are in extended markup language (XML) format. For each report, we remove all XML tags and use all text besides the identifier tags (registry ID, patient ID, tumor number, and document ID). Depending on the registry and the report, line breaks may or may not have been added to the body text to improve readability. In reports with line breaks, the line breaks are used to both demarcate segments of information (e.g. line break after each tumor property) as well as to break apart paragraphs of information after a set line length.

In cancer pathology reports for breast cancer, an important indicator for primary site are clock-time references (e.g. 12 o'clock or 03:00) that indicate the location of the tumor on the breast [98]. However, there is very little consistency in the format for clock-time across the corpus. To improve consistency across reports, we standardized all references to clock time to a single format – a number 1-12 followed by the string 'oclock'.

Table 2.1: Distribution of labeled reports for the primary site classification task and the histological grade classification task.

ICD-O-3 Topographical Codes		
Code	**Count**	**Description**
C34.0	26	Main Bronchus
C34.1	139	Upper lobe, lung
C34.2	11	Middle lobe, lung
C34.3	78	Lower lobe, lung
C34.9	191	Lung, NOS
C50.1	13	Central portion of breast
C50.2	36	Upper-inner quadrant of breast
C50.3	10	Lower-inner quadrant of breast
C50.4	63	Upper-outer quadrant of breast
C50.5	21	Lower-outer quadrant of breast
C50.8	62	Overlapping lesion of breast
C50.9	292	Breast NOS
Histological Grades		
G1	124	Well differentiated (low grade)
G2	233	Moderately differentiated (intermediate grade)
G3	271	Poorly differentiated (high grade)
G4	17	Undifferentiated (high grade)

In addition, we set all alphabetical characters to lowercase and remove all non-alphanumeric symbols except for periods, colons, and semicolons, as these symbols are used to split long lines. To ensure that periods are only used to mark the end of sentences, several additional preprocessing steps are taken to remove periods from abbreviations and floats; these steps are described in detail in Appendix A.2. Lastly, we tokenize the final text.

2.4.2 Word Embeddings

We use word embeddings to convert word tokens in each report into numerical vectors. Unlike simpler methods for representing words in vector form such as bag of words and one-hot encoding, word embeddings represent the semantic meaning of words within the numerical vectors [94]. Words that are semantically similar in meaning are closer to each other in distance, while words that are semantically different in meaning are further apart in distance. We evaluate two popular word embeddings–Word2Vec [93] and GloVe [110].

Word2Vec is based on the premise that words that appear in the same context tend to be more semantically similar. Word2Vec creates word embeddings by using a feedforward neural network to predict the neighboring words for a given word. Like Word2Vec, GloVe also uses word co-occurrence, but rather than predicting word context, GloVe is based on the idea that the ratio of the co-occurrence probabilities of two words can be used to embed semantic meaning for those words. We test both Word2Vec and GloVe embeddings during hyperparameter optimization as well as different sizes for the word embedding vector.

We train word embeddings on our entire corpus of 2505 reports and convert each token to its corresponding embedding. This converts each report into a tensor of size $\mathcal{X}_{L\times w\times e}$, where L is the number of lines per report, w is the number of words per line and e is the word embedding size. These tensors become the input into the hierarchical attention network (Appendix A.1).

2.4.3 Hierarchical Attention Network

Hierarchical Attention Networks (HANs) are a deep learning document classification model developed by Yang et al [152] based on recurrent neural networks. They are composed of

hierarchies in which the outputs of the lower hierarchies become the inputs to the upper hierarchies. This is based on the intuition that documents are composed of meaningful sequences of sentences, which are composed of meaningful sequences of words. Originally designed to classify reviews, HANs process a document one sentence at a time [152]. However, due to the structure and content of our pathology reports, the reports do not split naturally into sentences. We instead split each pathology report in our dataset by line breaks. To prevent long lines from reports without line breaks, we further split any line longer than 50 words by punctuation (periods, colons, and semicolons). This results in an average of 57 lines per document and 8 words per line for the entire corpus (Appendix A.3).

Each hierarchy in the HAN is composed of a dynamic bidirectional long short-term memory (LSTM) or gated recurrent unit (GRU) with attention mechanisms (Figure 2.1). Bidirectionality is imposed so that the network can account for the preceding and following context when processing words/sentences [82]. LSTMs/GRUs are used because they allow the network to selectively process input information based on how relevant it is to the classification tasks; likewise, the attention mechanism is added to enable the network to give extra focus on the LSTM/GRU outputs associated with the words and lines that are most indicative of a particular class. The implementation details of our LSTMs, GRUs, and attention mechanism are available in our Appendix A.4 and A.5. LSTMs and GRUs generally have similar performance on most tasks, but in specialized tasks one may outperform the other [48]. We therefore test both LSTMs and GRUs during hyperparameter optimization.

We model our HAN as two hierarchies. The lower hierarchy processes one line at a time, fed in as word embeddings. These are processed by a bidirectional LSTM/GRU with an attention mechanism that determines which words are most important. The output is a line embedding that captures the semantic content of the line. This is repeated for every line in a document. The upper hierarchy processes an entire document at a time by taking in the line embeddings generated from the lower hierarchy. These are fed into another bidirectional LSTM/GRU with an attention mechanism, all of which have the same architecture as those in the lower hierarchy. The final output is a weighted document embedding that represents the meaning of the entire document. We apply dropout to this final document embedding and then feed it into a softmax classifier to predict a class label.

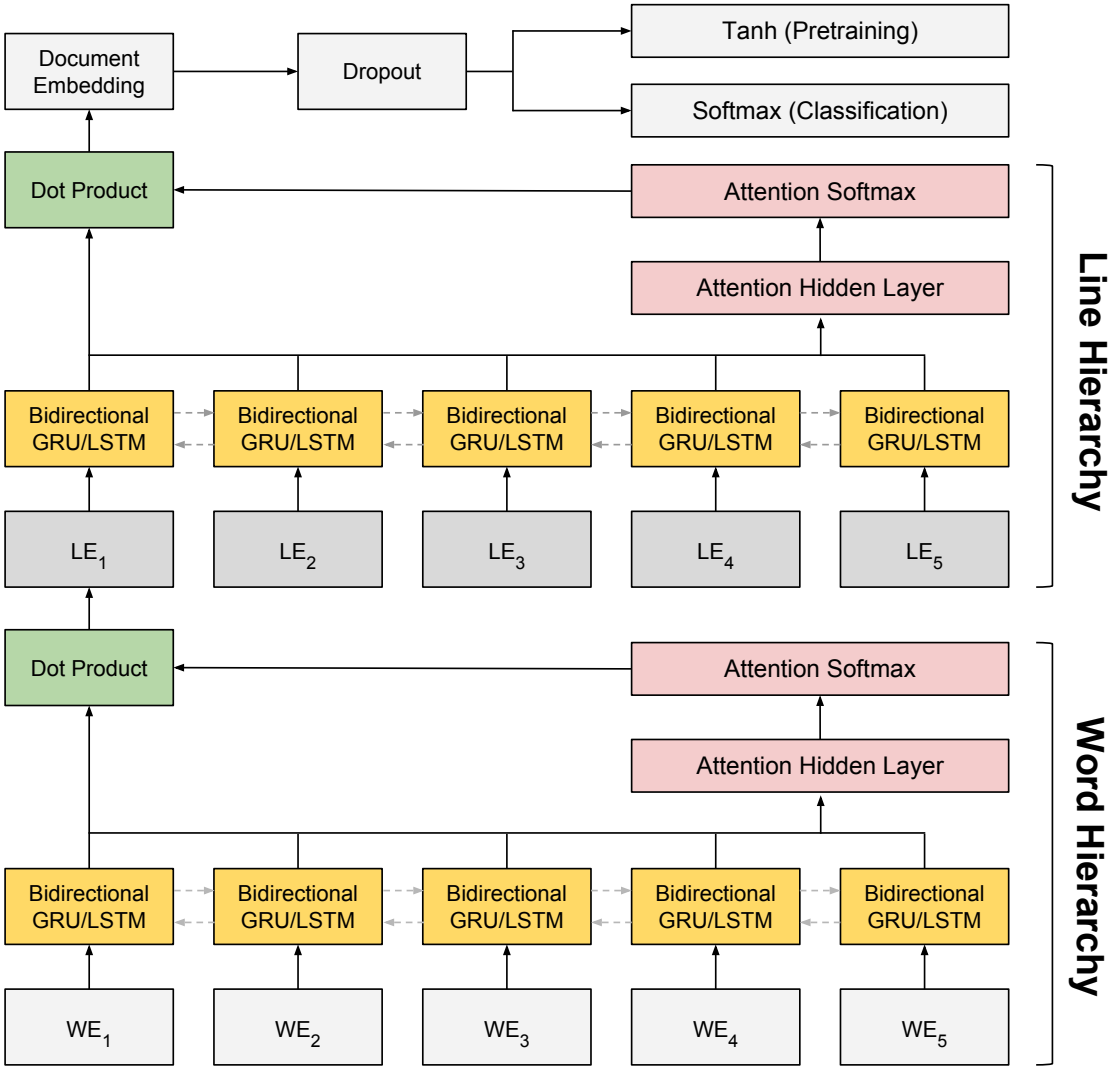


Figure 2.1: Architecture for our Hierarchical Attention Network (HAN). The HAN first produces line embeddings by processing the word embeddings in each line. The HAN then produces a document embedding by processing all the line embeddings in the document. The final document embedding can then be used for classification or pre-training purposes.

2.4.4 Hyperparameter Optimization

We use sequential optimization using gradient boosted trees to find the best hyperparameters for our hierarchical attention network. This optimization method uses a gradient boosted tree-based regression model to predict the model performance at unexplored hyperparameter settings. We use this optimization method because tree-based optimization has been shown to converge faster than traditional Bayesian optimization [9]. The following hyperparameters are tuned: (1) Type of word embeddings (Word2Vec or Glove); (2) Size of word embeddings (100-500); (3) Type of RNN unit used (GRU or LSTM); (4) Number of hidden GRU or LSTM cells used in each bidirectional RNN layer (50-200); (5) Size of hidden layer in attention mechanism (50-300) and (6) Dropout on final document embedding (0.5 or None). We run hyperparameter optimization separately for both the primary site classification task and the histological grade classification task to find the best architecture for each task. We optimize our hyperparameters on a validation set; our procedure is described in detail in Appendix A.6. The best hyperparameter setup for each classification task is listed in Appendix A.6.

2.4.5 Unsupervised Pretraining Step

Pretraining neural networks on unlabeled data using a modified autoencoder structure can improve the final performance of the network [37]. In these methods, the classification functions in the network are replaced with a decoder network that reconstructs the input data, therefore allowing the network to learn inherent patterns in the data. While such autoencoder structures may be effective for short text segments, it is far more difficult for an autoencoder to accurately regenerate long documents such as pathology reports[78].

We implement a novel method for pretraining our HAN using unlabeled reports: we model the final layer before the softmax classification as a document embedding to succinctly represent the content within a document. We can pretrain the network to learn inherent patterns in documents by having the network attempt to learn document embeddings that match some other unsupervised representation of document content. In our case, we train our model to generate document embeddings to match the term-frequency-inverse-document-frequency (TF-IDF) weighted word embeddings of the corresponding document, as we found

that these embeddings manage to capture some of the natural variation between documents of different classes (Appendix A.7).

To pretrain our model, we first generate TF-IDF vectors on unigrams only for all non-empty pathology reports from the full corpus – a total of 2495 reports. We then normalize each TF-IDF vector to have a total sum of 1. We create TF-IDF weighted word embeddings for a document by multiplying each index in that document's normalized TF-IDF vector with its corresponding word embedding and then summing the total. We normalize the resulting TF-IDF weighted word embedding to have mean 0 and standard deviation 0.4, then clip any values above 1 or below -1.

In our HAN, we replace the softmax classification layer with a tanh layer that attempts to output the corresponding TF-IDF weighted word embedding for the document (Eqn. 2.1, Figure 2.1).

$$pred_i = \tanh(W_{pred} emb_i + b_{pred}) \tag{2.1}$$

The model is trained for 5 epochs using mean-square-error loss on all non-empty pathology reports from the full corpus (Eqn. 2.2), and the trained weights are saved and used to initialize all model weights during supervised classification.

$$MSE = \frac{1}{n}\sum_{i}^{n}(pred_i - TFIDF_i)^2 \tag{2.2}$$

2.4.6 Experimental Setup and Evaluation Metrics

Our two classification tasks are primary site classification and histological grade classification. For both tasks, we use 10-fold cross validation to evaluate model performance. For each fold, we split the dataset into 0.8/0.1/0.1 train/validation/test sets. We train the HAN on the train set for 30 epochs and measure the performance on the validation set at each epoch. We save the network architecture at the epoch with the highest validation performance, and then evaluate final model performance on the test set.

Because of class imbalance, we use F-score as our evaluation metric for model performance. On the validation set, we chose the epoch with the highest micro F-score, which is a weighted average of the F-score for each class label weighted by class size (Eqn. 2.3-2.5).

$$P^{micro} = \frac{\sum_{C_j}^{C} TP_j}{\sum_{C_j}^{C} (TP_j + FP_j)} \tag{2.3}$$

$$R^{micro} = \frac{\sum_{C_j}^{C} TP_j}{\sum_{C_j}^{C} (TP_j + FN_j)} \tag{2.4}$$

$$F^{micro} = \frac{2P^{micro}R^{micro}}{P^{micro} + R^{micro}} \tag{2.5}$$

For final model performance, in addition to measuring micro F-score, we also measure macro F-score, which simply averages the F-score of each class label regardless of class size (Eqn. 2.6).

$$F^{macro} = \frac{1}{|C|} \sum_{C_j}^{C} F(C_j) \tag{2.6}$$

2.4.7 Model Baseline Comparisons

We compare the performance of the hierarchical attention network against the performance of several other predictive models that can be used for text classification. These include traditional machine learning classifiers as well as deep learning based classifiers. For traditional machine learning classifiers, we test Naive Bayes, Logistic Regression, Support Vector Machines, Random Forests, and XGBoost. We create input features for these traditional classifiers by removing punctuation and stop words from the pathology reports and then generating TF-IDF vectors on the resulting unigrams and bigrams with a minimum document frequency of three. To tune each classifier, we apply the same hyperparameter tuning with gradient boosted trees that was used on the HAN on each of the traditional machine learning classifiers. The final hyperparameters used for each classifier are listed

beneath the classifier name in Table 2.2. All classifiers were tested in Python using the `scikit-learn` package.

We also compare the performance of our HAN against the performance of recurrent neural networks (RNNs) and convolutional neural networks (CNNs). For our comparison, we use the same word embedding vectors as we use for the HAN. Unlike the HAN, which takes in a document one sentence at a time, both the RNN and the CNN take in all word embeddings in the entire document at once.

We test the performance of two RNN architectures. The first architecture is a RNN without an attention mechanism. This is equivalent to a single hierarchy of the HAN without the attention mechanism–a bidirectional RNN followed by dropout and softmax. The second architecture is an RNN with attention. This is equivalent to a single hierarchy of the HAN with the attention mechanism–a bidirectional RNN with attention followed by dropout and softmax. For both RNNs, we use the same optimized hyperparameters from our HAN hyperparameter optimization for each classification task. These baselines are designed to demonstrate how the attention mechanism and hierarchical structure of the HAN improve classification performance over basic RNNs. Furthermore, to confirm the necessity of the attention mechanism within both hierarchies of the HAN, we tested the performance of the HAN without the attention mechanism in either the word hierarchy or the line hierarchy.

For our CNN architecture, we use three parallel convolutional layers of 100 channels each with window sizes of three, four, and five words and stride of one word. For a given document, this results in an output of 300 channels x number of words. We then apply a maxpool on each of the 300 channels across all words in the document, resulting in a vector of length 300 for each document. A dropout of 50% is applied to this vector, which is finally fed into a softmax classifier. This particular architecture has been shown to perform well on cancer pathology report classification [117].

To maintain consistency, all models were trained using the same 0.8/0.1/0.1 train/ validation/test splitting as the HAN. All deep learning models were trained for 30 epochs, and the model weights during the epoch with the highest validation performance were used for test set evaluation.

2.5 Experimental Results

2.5.1 Primary Site Classification

In the primary site classification task, a predictive model must predict the primary site of a cancer given the content of the cancer pathology report. The performance results on this task are displayed in Table 2.2, which includes 95% confidence intervals for each performance metric derived using bootstrapping [29]. The bootstrapping method is described in greater detail in Appendix A.8.

We see that the HAN outperforms all other models in both the micro F-score metric with a score of .800 and the macro F-score metric with a score of .594. This is an improvement of 13% for micro F-score over the next best model, convolutional neural networks, and an improvement of 27% for macro F-score over the next best model, recurrent neural networks. Pretraining the network using TF-IDF weighted word embeddings yields increases in both the speed of convergence of the model and the final performance accuracy (Fig. 2.2).

2.5.2 Histological Grade Classification

In the histological grade classification task, our model must predict the histological grade of a cancer given the cancer pathology report. The performance results on this task are displayed in Table 2.2, which includes 95% confidence intervals for each performance metric derived using bootstrapping [29].

Like in the primary site classification task, the hierarchical attention network once again outperforms all other models in both the micro F-score metric with a score of .916 and the macro F-score metric with a score of .841. This is an improvement of 28% for micro F-score over the next best model, convolutional neural networks, and an improvement of 37% for macro F-score over the next best model, XGBoost. While pretraining the network using TF-IDF weighted word embeddings yields an initial increase in both the speed of convergence and the validation accuracy (Figure 2.2), pretraining does not boost the final test accuracy in the histological grade classification task. We believe this is because TF-IDF weighted

Table 2.2: Final test performance of classification models on each classification task. Classifier performance and confidence intervals on individual tasks are shown within parentheses.

Classifier	Primary Site Micro F-Score	Primary Site Macro F-Score	Hist. Grade Micro F-Score	Hist. Grade Macro F-Score
Traditional Machine Learning Classifiers				
Naive Bayes	.554 (.521, .586)	.161 (.152, .170)	.481 (.442, .519)	.264 (.244, .283)
Logistic Regression (penalty = l1, solver = liblinear, C = 3.3, multiclass = ovr)	.708 (.678, .737)	.400 (.361, .437)	.657 (.622, .693)	.507 (.433, .584)
Support Vector Machine (C = 25, kernel = sigmoid, gamma = 0.5, shrinking = True)	.673 (.643, .702)	.396 (.353, .435)	.595 (.558, .634)	.472 (.413, .540)
Random Forest (num trees = 400, max features = 0.9)	.701 (.673, .730)	.437 (.406, .467)	.694 (.657, .727)	.579 (.503, .650)
XGBoost (max depth = 5, num trees = 300, learning rate = 0.3)	.712 (.683, .740)	.431 (.395, .466)	.681 (.643, .716)	.612 (.539, .673)
Deep Learning Classifiers				
Convolutional Neural Network	.712 (.680, .736)	.398 (.359, .434)	.716 (.681, .750)	.521 (.493, .548)
Recurrent Neural Network (without attention mechanism)	.617 (.586, .648)	.327 (.292, .363)	.393 (.353, .431)	.275 (.245, .304)
Recurrent Neural Network (with attention mechanism)	.694 (.666, .722)	.468 (.432, .502)	.580 (.541, .617)	.474 (.416, .536)
Hierarchical Attention Network (no pretrain, word attention only)	.695 (.666, .725)	.405 (.367, .443)	.473 (.437, .512)	.341 (.302, .390)
Hierarchical Attention Network (no pretrain, line attention only)	.731 (.704, .760)	.464 (.425, .503)	.473 (.434, .512)	.340 (.301, .388)
Hierarchical Attention Network (no pretrain, word and line attention)	.784 (.759, .810)	.566 (.525, .607)	**.916 (.895, .936)**	**.841 (.778, .895)**
Hierarchical Attention Network (pretrain, word and line attention)	**.800 (.776, .825)**	**.594 (.553, .636)**	.904 (.881, .927)	.822 (.744, .883)

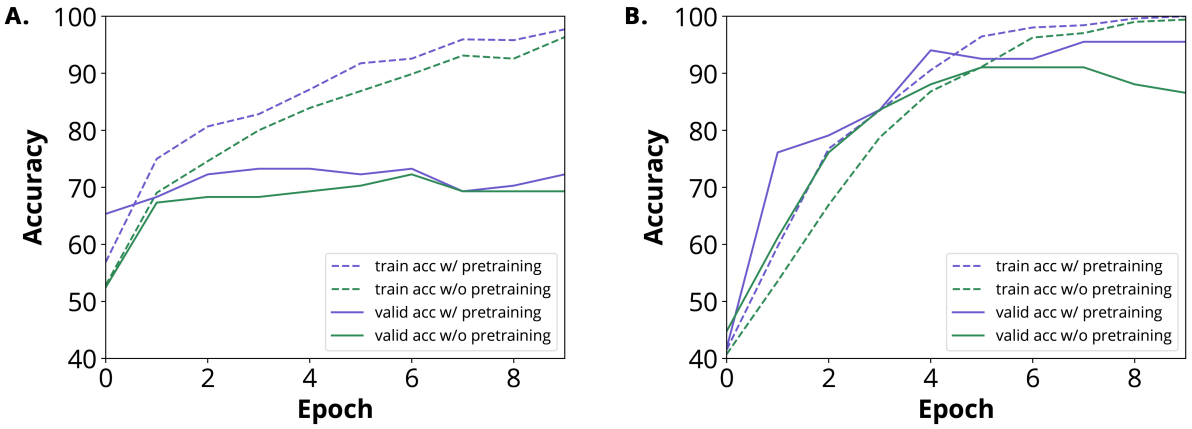


Figure 2.2: HAN train and validation accuracies with and without pretraining during first 10 epochs for (A.) primary site classification task and (B.) histological grade classification task.

word embeddings do not capture the natural separation of classes for histological grades as well as for primary sites (Appendix A.7).

2.5.3 Model Visualizations

The HAN can provide additional insight about the corpus of pathology reports by utilizing the structure of its architecture. Given a document, we can use the attention mechanism weights in both the word and line hierarchies to visualize how much each word and each line contributes to the final classification of that document (Figure 2.3). We can also find the words in our vocabulary associated with the highest attention weights to identify the words that are the most important to each classification task (Appendix A.11).

In addition, because the final layer of the HAN creates a document embedding for each pathology report, we can use a dimensionality reduction technique such as PCA to plot the similarity of the pathology reports in our corpus based on their corresponding document embeddings (Figure 2.4). From this plot, we can see the natural clustering that occurs between pathology reports with similar cancers. We can also use this technique to locate and better understand misclassified reports based on the location of their document embeddings relative to existing document clusters. We note that the document embeddings generated by the HAN segments the different classes significantly better than unsupervised techniques such as TF-IDF weighted word embeddings (Appendix A.7) and Doc2Vec (Appendix A.9). The HAN embeddings are also better segmented than document embeddings generated by the CNN (Appendix A.10).

2.5.4 Error Analysis

We analyzed the confusion matrices for the HAN in both the primary site and histological grade classification tasks (Figure 2.5). In the primary site classification task, we can see that the vast majority of misclassifications were within the same type of cancer rather than between the two types of cancer. In other words, misclassified breast cancer sites were mostly misidentified as other breast cancer sites rather than as lung cancer sites, and vice versa.

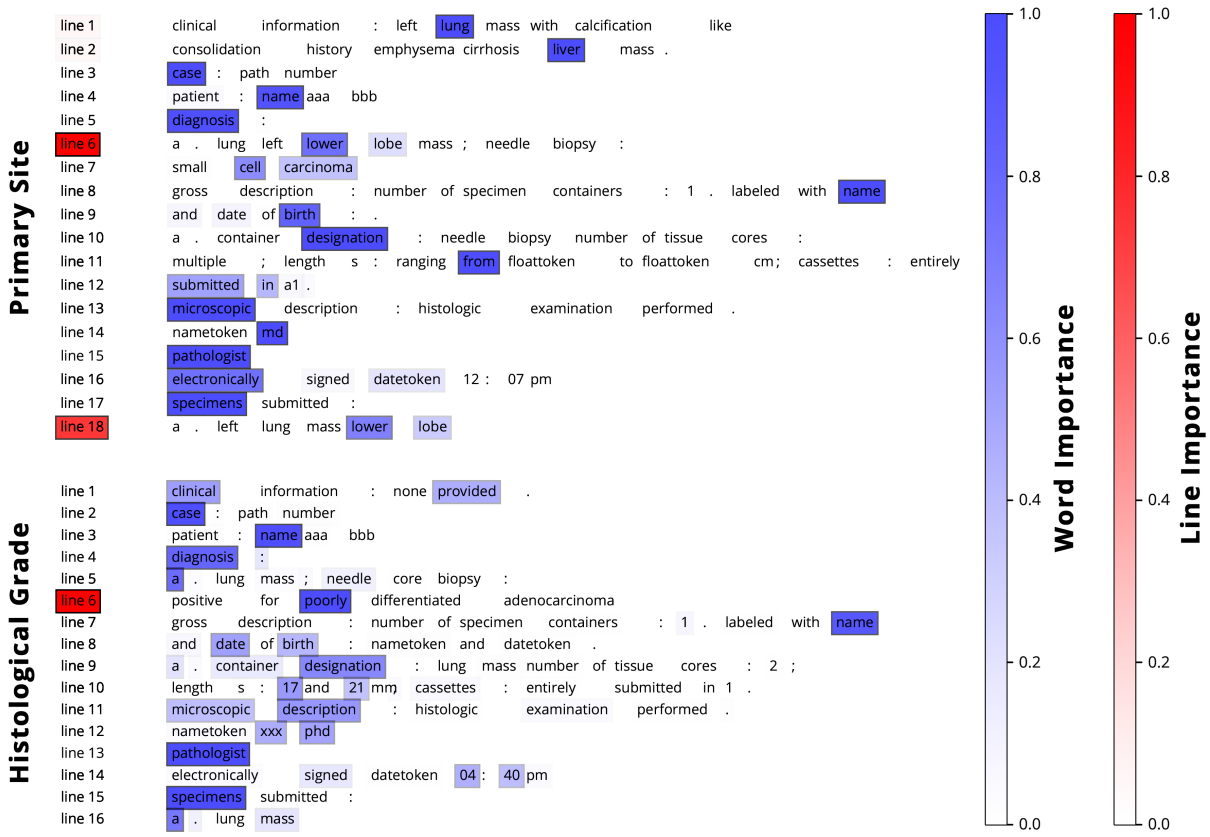


Figure 2.3: HAN annotations on sample pathology report for each classification task. The most important words in each line are highlighted in blue, with darker blues indicating higher importance. The most important lines in the report are highlighted in red, with darker red indicating higher importance. For each task, the HAN can successfully locate the specific line(s) within a document and text within that line that identify the primary site (e.g., lower lobe) or histological grade (e.g., poorly differentiated). The RNN structure utilized by the HAN allows it to take into account word and line context to better locate the correct text segments.

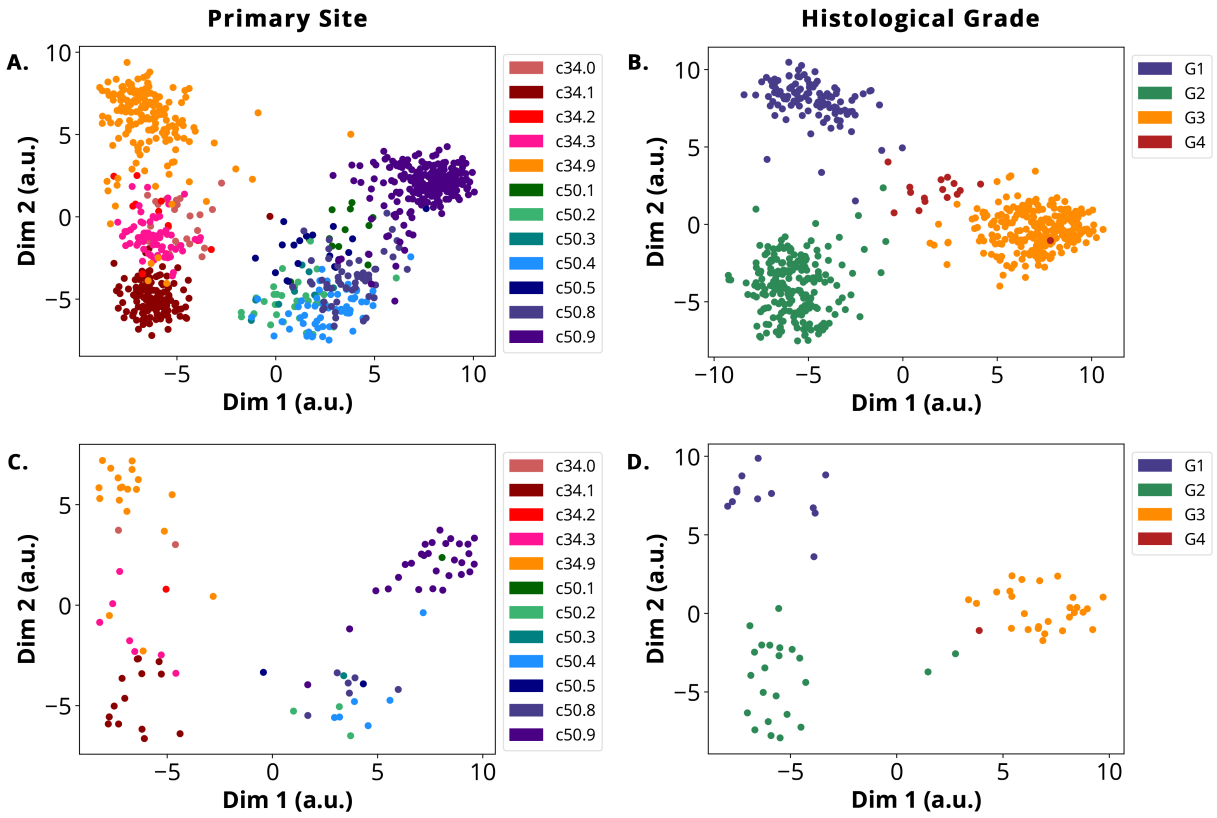


Figure 2.4: HAN document embeddings reduced to 2-dimensions via PCA for (A.) primary site train reports, (B.) histological grade train reports, (C.) primary site test reports, and (D.) histological grade test reports.

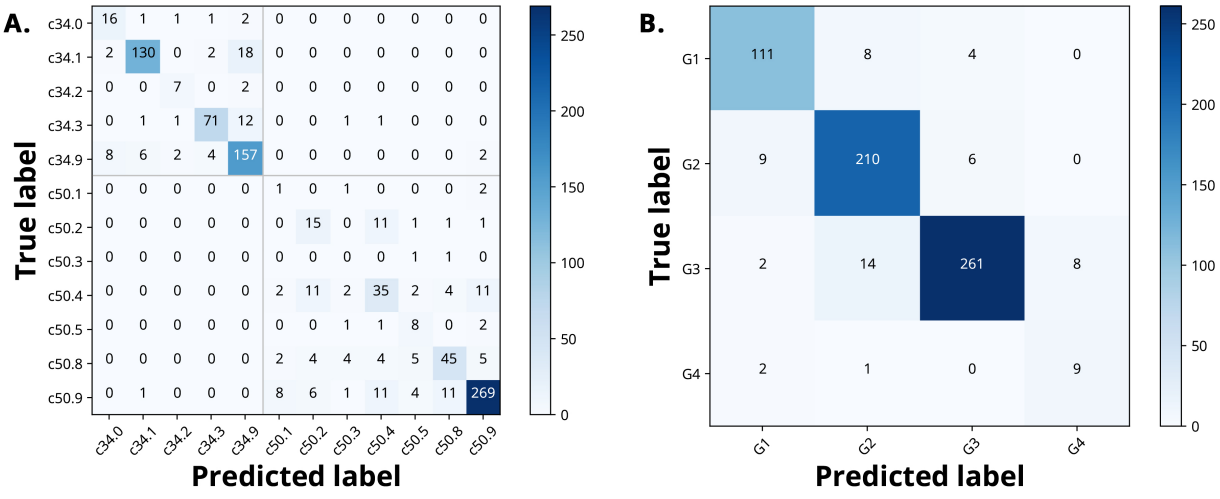


Figure 2.5: Confusion matrix for (A.) HAN w/ pretraining on primary site classification task and (B.) HAN w/o pretraining on histological grade classification task.

As expected, the sites with the highest number of samples had the highest individual F-scores, while the sites with the lowest number of samples had the lowest F-scores. The largest proportion of misclassification were related to the labels c34.9 and c50.9, which are respectively used to label lung cancer reports where the subsite is not specified and breast cancer reports where the subsite is not specified. In these reports, the HAN may have mistakenly identified features suggesting a particular subsite. Another label with a high number of misclassifications is c50.8, which is used to identify reports for breast cancer that overlaps several subsites. In these reports, the HAN may have identified only one of the subsites and failed to recognize that several subsites were involved. We note that the HAN has stronger performance dealing with the ambiguities in labels c34.9, c50.8, and c50.9 compared to the other classification models based on the confusion matrices for each model (Appendix A.13).

In the histological grade classification task, G4 was the least populated with only 17 samples and therefore had the highest proportion of misclassifications. Among the other three grades, the HAN maintains strong performance.

2.6 Discussion and Conclusions

Based on the results in Table 2.2, we have shown that HANs can classify pathology reports better than traditional machine learning classifiers that use TF-IDF features. Because vector space models such as TF-IDF and bag-of-words cannot take into account the ordering of words beyond a small window, machine learning classifiers that rely on these representations can only make decisions based on the appearance of specific word tokens irrespective of their context. Cancer pathology reports (and other unstructured text documents) typically include complex information spread over long segments of text, and our results suggest that vector space models are unable to effectively capture the deeper and more abstract linguistic patterns required for higher level information abstraction.

On the other hand, HANs do not suffer as much from this shortcoming because their structure is designed to capture meaningful linguistic patterns across longer sequences of words within a sentence and sentences within a document. The use of RNNs within the HAN

structure enable the HAN to better distinguish the nuances between the same words or word groups across different contexts. Furthermore, the HAN (and other deep learning models) rely on word embedding representations of words rather than the word tokens themselves; because these word embedding representations capture word similarity, this allows the HAN to better process never-before-seen word sequences based on their similarity to existing words and sequences. The high performance of the HAN suggests that the combination of these approaches is better at capturing important information than making decisions based on simply the appearance of specific words or groups of words.

Our results also show that HANs outperform other deep learning architectures. Like in TF-IDF based models, convolutional neural networks are limited in that they can only process a maximum window of (five in our implementation); any linguistic context beyond this window cannot be captured. RNN-based networks can process much longer word segments, allowing them to capture meaningful linguistic abstractions across longer contexts of words. However, basic RNNs have shortcomings in that they can have difficulty retaining important information over a very large number of timesteps [20]. While an attention mechanism mitigates this issue, our results show that attention mechanisms are not effective enough for long documents such as pathology reports.

HANs address these issues by taking into account the natural segmentation of text. The HAN first creates a representation of each sentence in a document based off the most important words in that sentence; it then creates a representation of the entire document based on the most important sentences in the document. By breaking down long documents into smaller chunks, the HAN can better locate and extract critical information. Our results suggest that this approach works more effectively than other artificial methods for processing text.

By examining the internal weights of the HAN, we can visualize task-specific words/lines that enable one to understand how the pathology reports are being processed. These visualizations, apart from confirming that the HAN is learning the correct representations for a document, can enable clinicians to further analyze and annotate pathology reports for additional findings. In particular, the HAN can first identify the segments in a pathology report relevant to a specific classification task, and then a human expert can follow up and

use the annotations from the HAN in other related tasks. We posit that this represents a first step towards automated annotation of clinical texts.

Pretraining the HAN provides an additional performance boost on the primary site classification task by allowing the HAN to first learn the inherent structure of a corpus before further tuning the weights on a fine-grained classification task (Figure 2.2). This is especially useful because many pathology reports available in cancer registries are unlabeled. Our pretraining approach allows the HAN (and other similar deep learning models) to benefit from these reports without requiring a trained expert to manually create labels first [37]. A natural extension of the pretraining is to use these learned features as priors for semi-supervised models [62, 63].

A study by Powsner and colleagues showed that surgeons misunderstand pathology reports up to 30% of the time [115]. With our method, we aim to reduce both the time required and error rate for pathology report classification. In the primary site classification task, our HAN achieves micro F-score performance levels of 0.80. We believe that performance could be improved even further if not for the inherent ambiguity in the labels for three classes namely c34.9, c50.8, and c50.9, which represent documents that were not annotated with a subsite or were annotated with multiple subsites. In the histological grade classification task where this ambiguity did not exist, the HAN achieved micro F-score performance levels of 0.92. Furthermore, the HAN's performance was limited by the relatively small size of our dataset–for several classes we had only tens of samples for the HAN to train on. We believe that provided with a larger dataset, the HAN can better learn the linguistic patterns in the reports and achieve levels of performance that surpass that of human experts, enabling the HAN to play an effective and practical role in the automation of cancer pathology report classification.

Due to the complexity of the HAN architecture relative to other machine learning architectures, the HAN takes longer to process documents compared to the other models tested. However, this limitation does not preclude practical application of the HAN for document classification. On a single CPU, the HAN can still process the entire corpus of 2505 documents in approximately 150 seconds (Appendix A.12). We expect that with the emergence of newer computer architectures and better training approaches for RNNs, both

the training and testing time performance can be significantly improved. Analyses of scaling and performance behaviors of our implementation is currently underway.

One important limitation of our study is the small size of our dataset. Because deep learning models learn their own features from raw data, they often require large datasets containing thousands or millions of samples to reach full effectiveness. Since our dataset used for supervised training consists of less than 1000 samples, our deep learning approaches risk overfitting on the data. Given a larger dataset, we expect the effectiveness of the HAN relative to other approaches to improve even further. Yang et al. demonstrated that the HAN significantly outperforms CNNs, RNNs, and other traditional machine learning classifiers on large public datasets such as Yelp and IMDB [152], and we expect the same to hold true for our pathology reports and other biomedical datasets. To confirm this, we tested the performance of the HAN on a simple 8-class topic classification task with Pubmed abstracts. We found that increasing the number of abstracts from 8,000 to 80,000 to 800,000 yielded overall classification accuracies of 70.63%, 76.50%, and 76.88%; additional details are available in Appendix A.14. We also have plans to run our pathology report classification experiments on a much larger dataset of 20,000 labeled pathology reports that was unavailable during the time of this study.

Moving forward, we will extend the HAN to perform simultaneous classification on multiple tasks. Deep learning models adapted for multitask learning have been shown to have higher performance on subtasks compared to the same models adapted for single-task classification [155]. By adapting the HAN to simultaneously predict both primary site and histological grade, we expect to be able to increase performance in both tasks. Another promising area of research lies in automating the generation of pathology reports and other medical documents from raw imaging data or other raw medical data sources. Like the decoding process for a cancer pathology report, the encoding process of tumor data into a cancer pathology report is subject to human error and variability in the interpretation of coding guidelines. An effective automated solution could standardize how tumor data is encoded into pathology reports and yield improvements in the accuracy and efficiency of the report processing pipeline.

The Python code developed based on TensorFlow is available at http://code.ornl.gov/v33/PathRepHAN. Enhancements for multiple graphics processing unit (GPU)-based training and performance/scaling behaviors will be documented as part of further development.

Chapter 3

Classifying Cancer Pathology Reports with Hierarchical Self-Attention Networks

A version of this chapter has been accepted for publication in the journal Artificial Intelligence in Medicine:

Shang Gao, John X. Qiu, Mohammed Alawad, Jacob D. Hinkle, Noah Schaefferkoetter, Hong-Jun Yoon, Blair Christian, Paul A. Fearn, Lynne Penberthy, Xiao-Cheng Wu, Linda Coyle, Georgia Tourassi, Arvind Ramanathan. "Classifying Cancer Pathology Reports with Hierarchical Self-Attention Networks." Artificial Intelligence in Medicine, 2019.

Shang Gao, Arvind Ramanathan, and Georgia Tourassi conceptualized the study and developed the experimental setup. Shang Gao designed and implemented the HiSAN and ran the experiments. Shang Gao, John Qiu, Mohammed Alawad, Jacob Hinkle, Noah Schaefferkoetter, Hong-Jun Yoon, and Blair Christian developed the text preprocessing and experimental pipeline. Paul Fearn, Linda Coyle, Xiao-Cheng Wu and Linda Coyle oversaw and organized the preparation of the pathology reports. Shang Gao wrote the manuscript with support from Arvind Ramanthan and Georgia Tourassi. All authors read and approved the manuscript.

No revisions to this chapter have been made since the original publication.

3.1 Abstract

We introduce a deep learning architecture, Hierarchical Self-Attention Networks (HiSANs), designed for classifying pathology reports and show how its unique architecture leads to a new state-of-the-art in accuracy, faster training, and clear interpretability. We evaluate performance on a corpus of 374,899 pathology reports obtained from the National Cancer Institute's (NCI) Surveillance, Epidemiology, and End Results (SEER) program. Each pathology report is associated with five clinical classification tasks – site, laterality, behavior, histology, and grade. We compare the performance of the HiSAN against other machine learning and deep learning approaches commonly used on medical text data – Naive Bayes, logistic regression, convolutional neural networks, and hierarchical attention networks (the previous state-of-the-art). We show that HiSANs are superior to other machine learning and deep learning text classifiers in both accuracy and macro F-score across all five classification tasks. Compared to the previous state-of-the-art, hierarchical attention networks, HiSANs

not only are an order of magnitude faster to train, but also achieve about 1% better relative accuracy and 5% better relative macro F-score.

3.2 Introduction

The National Cancer Institute's (NCI) Surveillance, Epidemiology, and End Results (SEER) program maintains repositories of detailed cancer pathology reports of the majority of cancer cases collected by its registries throughout the United States (https://seer.cancer.gov) – these reports represent a rich source of information on detailed cancer characteristics. However, extracting useful information such as cancer site or histology from pathology reports is usually done manually by specialists who are trained with specific knowledge for understanding the contents of the reports. This process is very costly and time-consuming, thereby limiting the ability of cancer registries to identify useful data elements from existing pathology reports and restricting time-sensitive applications such as precision medicine. Therefore, there is an immediate need to modernize the information extraction process across all cancer registries.

Automated classification of cancer pathology reports is an active area of research that aims to utilize machine learning (ML) methods to identify key data elements from pathology reports, thereby reducing the burden on human annotators. Recently, deep learning approaches have been shown to be greatly effective for various natural language processing (NLP) tasks, such as parsing [4], sentiment analysis [158], and question answering [52]; as a result, many of these approaches have been adapted for medical NLP tasks, including clinical text classification[96] and information extraction [44].

Unfortunately, these deep learning architectures are generally designed for shorter text segments typically no longer than one paragraph. Cancer pathology reports present unique challenges including documents that are several pages long, long-distance linguistic dependencies over different report sections, and only a tiny fraction of the pathology report text being relevant to a specific classification task. Moreover, the variation in report structure and linguistic patterns across pathology labs present additional challenges. Therefore, many

out-of-the-box deep learning architectures adapted from general NLP tasks are unable to efficiently tackle these challenges.

In this work, we introduce a deep learning architecture, the Hierarchical Self-Attention Network (HiSAN), that is specifically designed to overcome many of the difficulties associated with cancer pathology reports. We test the effectiveness of the HiSAN against other state-of-the-art text classification methods on classifying five key cancer data elements – site, laterality, behavior, histology, and grade – from a dataset of approximately 375K cancer pathology reports. We show not only that the HiSAN achieves the best accuracy and macro F-score across all five tasks, but also that it runs more than an order of magnitude faster than similar hierarchical deep learning methods. Furthermore, we show how the HiSAN's unique architecture enables clear visualization of the specific keywords in each pathology report that identify each data element as well as visualization of the linguistic relationships between different words and text segments within a report.

3.3 Background

Tracking cancer statistics across the population of the United States is an important priority for the Centers for Disease Control and Prevention (CDC), the North American Association of Central Cancer Registries (NAACCR), the American Cancer Society (ACS), and the National Cancer Institute (NCI) [35]. These statistics help inform cancer research, funding, and legislation, identify populations at risk for certain cancer types, and analyze the impact of early detection and treatment advances. NCI's SEER program maintains cancer registries across the US that collect, store, and manage cancer incidence and survival data covering approximately 34.6% of the US population (https://seer.cancer.gov/about/overview.html); detailed cancer characteristics such as anatomic site and histology are generally recorded in the form of cancer pathology reports. Because manually parsing cancer pathology reports is a time-intensive task requiring expert knowledge, automated tools for pathology report IR can significantly ease the burden on human annotators.

Traditionally, most automated classification systems for pathology reports relied on rule-based approaches [14, 100, 145, 73, 148, 101, 24] or non-deep-learning ML approaches, such as

Naive Bayes, logistic regression, Adaboost, support vector machines, and conditional random fields [88, 80, 150, 154, 162, 144]. These approaches generally rely on either hand-engineered features or the appearance of specific n-gram word phrases to identify key data elements. These approaches have significant weaknesses – hand-engineered features can be brittle and fail when a document does not adhere to the pre-determined rules, while n-gram features are unable to account for any potentially important linguistic patterns that span beyond a short window of text [114, 160].

In recent years, many automated NLP solutions have shifted towards the use of deep learning, as deep learning approaches have been shown to beat out traditional ML approaches across a wide range of NLP tasks [156]. One important benefit of deep learning approaches is their ability to learn their own salient features directly from text without the need for any human engineering; furthermore, unlike approaches that utilize n-gram features, many deep learning architectures can identify relevant linguistic patterns across long spans of text. Moreover, deep learning approaches generally represent text as word embedding vectors, which capture the semantic content of words in vector format [13]. Because word embedding representations can capture linguistic meaning and word similarity, deep learning approaches are better able to generalize when applied to documents with new words that have never been seen by the model. Recent studies have shown that deep learning approaches can outperform traditional ML and rule-based approaches in various clinical NLP tasks [116, 147].

Convolutional neural networks (CNNs) are a popular type of deep learning architecture for NLP that have been effectively applied towards medical text analysis [55, 116]; these utilize a sliding window that moves across the document and analyzes a set number of words at a time. Like traditional ML models that utilize n-gram based features, one noticeable weakness of CNNs is that they are unable to capture linguistic patterns beyond a small window, usually 3-5 words. Attempts have been made to mitigate this weakness by stacking multiple convolution layers to increase the receptive field [25]; however, a comparative study showed that these deeper networks are often unable to outperform their shallower counterparts [72].

Recurrent neural networks (RNNs) are another popular type of deep learning architecture for NLP and medical text analysis [58, 59]. RNNs are designed to handle sequential data.

They process a word sequence one word at a time, and at each word, the RNN accounts for information from all previous words in the sequence. Unlike CNNs, RNNs can therefore find linguistic patterns over very long text sequences. However, because the operations associated with each word in a text sequence depend on all previous words, RNNs are very slow compared to CNNs and are also difficult to parallelize [107]. The computational complexity of an RNN scales polynomially with the length of a given sequence; therefore, while RNN-based approaches may be tractable for shorter text documents such as Yelp reviews, they can be prohibitively slow for longer, multi-page text documents such as pathology reports.

In our previous work, we analyzed the effectiveness of Hierarchical Attention Networks (HANs) [152], an RNN-based architecture, on the effectiveness of identifying the cancer site and histological grade from pathology reports and found that HANs beat out CNNs and traditional ML approaches on a small set of 1000 pathology reports [41]. However, we also found that HANs are very slow to train, requiring a processing time of nearly a second per report – training multiple epochs on a large corpus of pathology reports could mean several weeks of train time.

Self-attention is a relatively new development in the deep learning community that has achieved state-of-the-art performance in various NLP tasks, including machine translation [141] and question answering [157]. Self-attention-based approaches utilize neural attention mechanisms (discussed in detail in the following section) that find linguistic relationships between different words within a text sequence. These approaches offer the best of both worlds – like RNNs, self-attention can find linguistic patterns across very long text sequences, and like CNNs, self-attention is fast to run. Another major benefit of attention-based architectures is that they are easy to interpret and visualize – the outputs from a neural attention mechanism directly tell a user which words the model is using to make a decision.

In this work, we introduce the HiSAN, which utilizes a hierarchical architecture similar to the HAN but replaces the computationally expensive RNN layers with self-attention. We expand our previous experiments from two to five classification tasks and test on a much larger dataset of pathology reports. We show that HiSANs can not only achieve better accuracy and macro F-score than HANs but also train over ten times faster.

3.4 Materials and Methods

3.4.1 Dataset Details

Our full dataset consists of 374,899 cancer pathology reports obtained from the Louisiana tumor registry in May 2018 in accordance to the institutional review board protocol DOE000152. These reports cover all types of cancers among Louisiana residents from the years 2004 to 2018. Each report includes metadata, such as the date of the report and the unique patient ID, as well as the full clinical comments written by the pathologist analyzing the cancer tumor.

Each pathology report includes a unique tumor ID identifying the specific tumor associated with that report; multiple reports may be associated with the same tumor ID. The Louisiana cancer registry provided us with the ground truth labels for five key data elements – site, laterality, behavior, histology, and grade – associated with each unique tumor ID. These were manually annotated by a human expert based off all pathology reports associated with that tumor ID as well as other available data for that specific tumor, including clinical notes. There are a total of 70 possible labels for site, 7 for laterality, 4 for behavior, 516 for histology, and 9 for grade; the label descriptions and the number of occurrences per label are available in Appendix B.1, and more detailed information can be found in the SEER coding manual (https://seer.cancer.gov/tools/codingmanuals/).

To simulate a real production environment in which a classifier trained on older existing reports must predict labels for new incoming reports, we split our dataset into train, validation, and test sets based off date. Because the same tumor ID may have multiple pathology reports associated with it over time, we designed our splitting to prevent reports from the same tumor ID being split between the train, validation, and test sets. Therefore, we first grouped all pathology reports by tumor ID; each pathology report belonging to the same tumor ID is assigned the date of the earliest report associated with that tumor ID – a report written in 2017 may be assigned a 2012 date if it belongs with a tumor ID with a report first written in 2012. We isolated all reports from 2016 and later into our test set; from the remaining reports from 2004-2015, we randomly selected 80% for our train set and used the other 20% for our validation set (ensuring reports from the same tumor ID were not

split between both train and validation sets). This results in a train set of 236,519 reports, a validation set of 59,241 reports, and a test set of 78,856 reports.

Data Cleaning

Each raw pathology report was provided to us in XML format that included both metadata and text fields. For each report, we discarded the metadata fields, such as patient ID and registry ID, and retained all text fields, such as clinical history and formal diagnosis. We then lowercased all words, converted any unicode characters into their corresponding ascii characters, and removed any consecutive punctuation (e.g., multiple periods following one another were replaced by a single period). Any unique words appearing fewer than five times across the entire corpus were replaced with an "unknown_word" token.

We applied several text modification and replacement steps to standardize the pathology reports. These include standardizing all clock-time references, which are used to identify cancer site in cancers such as breast cancer, into the format of a number 1-12 followed by the string "oclock". In addition, to reduce the vocabulary space, we converted all decimals into a "decimal" word token and all integers larger than 100 into a "large_integer" word token. Additional minor text modification and replacement steps are listed in Appendix B.2. After cleaning, the average pathology report had a length of 633 tokens.

Like the HAN, the HiSAN first breaks a long document down into smaller linguistic segments, such as individual sentences. Generally speaking, pathology reports do not naturally break down into sentences; instead, most pathology reports list relevant facts and information line by line. We therefore split each document into smaller segments based off the natural linebreaks occurring in each pathology reports. Unfortunately, not all pathology reports use line breaks to separate information – in some cases, facts are presented in long paragraphs of natural language or are separated by symbols such as '#' or '<'. Consequently, after splitting pathology reports by linebreaks, if any line is longer than 50 words, we further split it based off a curated list of punctuation and symbols based off our observations of the characters used to itemize lists within our corpus; these symbols are provided in Appendix B.2. After splitting into lines, the average pathology report had 70 lines with an average of 8.5 tokens per line.

3.4.2 Hierarchical Self-Attention Networks

The architecture of the HiSAN is similar to that of the Hierarchical Convolutional Attention Network (HCAN) [40], which is an architecture we previously developed for sentiment analysis and general text classification tasks. In our experiments, we found that several components used in the HCAN which improved performance on general text such as Yelp and Amazon reviews instead reduced performance when applied to clinical text classification – a detailed ablation study is available in Appendix B.7. Therefore, we propose the new HiSAN architecture, which is better suited for classification of cancer pathology reports.

The structure of our HiSAN is shown in Figure 3.1. Each component of the HiSAN is discussed in greater detail in the following subsections.

Self-Attention

A self-attention mechanism compares a sequence of embeddings against itself to find relationships between the entries in the sequence. Given a sequence of embeddings $E \in \mathbb{R}^{l \times d}$, where l is the length of the sequence and d is the embedding dimension, a basic self-attention mechanism generates a new sequence $S \in \mathbb{R}^{l \times d}$ in which each entry s_i is a weighted average of all entries e_i in the original sequence. Intuitively speaking, each new entry s_i should have captured within it the most pertinent information to that entry from all entries e_i in the original sequence:

$$\text{Self-Attention}(E) = \text{softmax}(EE^\top)E \tag{3.1}$$

To improve upon this basic self-attention, rather than directly compare E against itself, we use functions to extract three different sets of features from E: (1) Q and (2) K, which are features that help find important relationships between entries in the sequence, and (3) V, which are the features that are used to generate the new output sequence. This allows for more expressive comparison between entries in a sequence. Certain features are highly useful when finding word relationships, such as identifying how biomedical terms correspond to each other in a pathology report, and these are captured in Q and K. On the other hand, certain features are more useful for the final classification task being targeted, and these are captured in V. We use three position-wise feedforward operations on the same

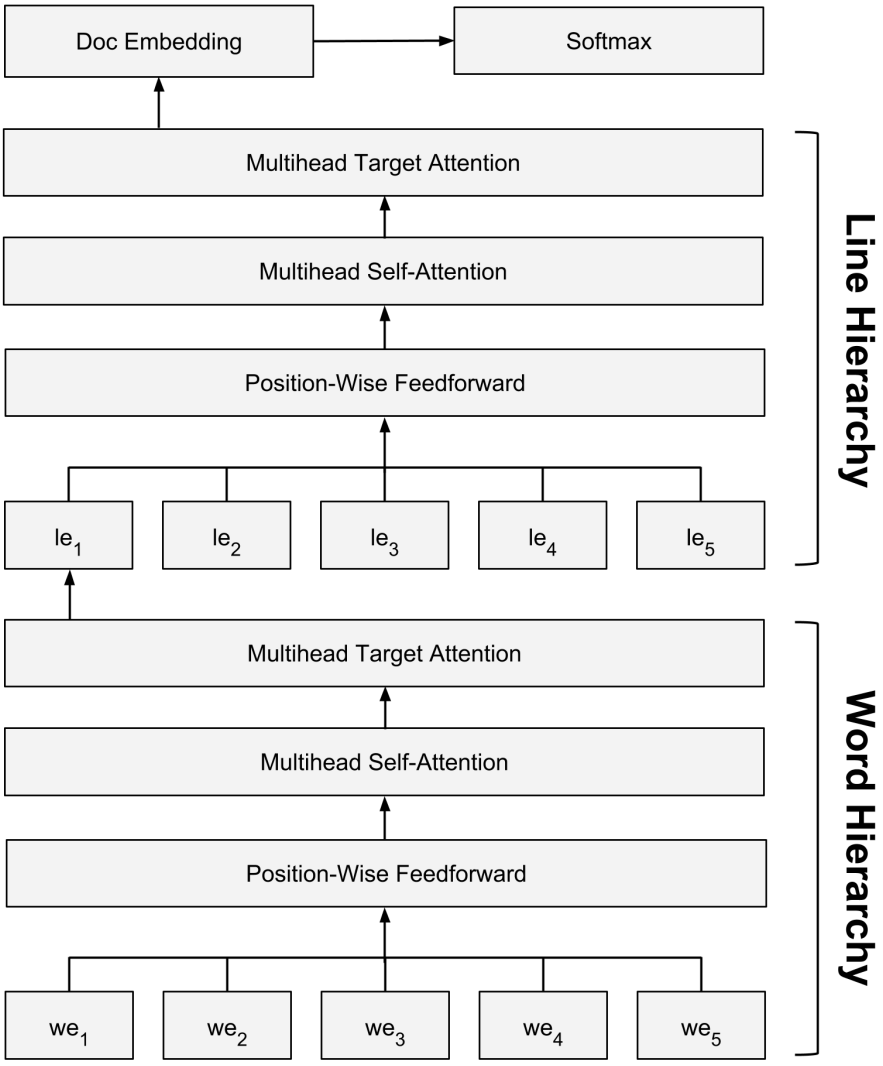


Figure 3.1: Architecture for our Hierarchical Self Attention Network (HiSAN).

input sequence $E \in \mathbb{R}^{l \times d}$ to generate $Q \in \mathbb{R}^{l \times d}$, $K \in \mathbb{R}^{l \times d}$, and $V \in \mathbb{R}^{l \times d}$. Our position-wise feedforward operation is equivalent to a 1D convolution operation with a window size of one word:

$$\begin{aligned} Q &= \text{ELU}(\text{Conv1D}(E, W^q) + b^q) \\ K &= \text{ELU}(\text{Conv1D}(E, W^k) + b^k) \\ V &= \text{ELU}(\text{Conv1D}(E, W^v) + b^v) \\ \text{Self-Attention}(Q, K, V) &= \text{softmax}(\frac{QK^T}{\sqrt{d}})V \end{aligned} \tag{3.2}$$

In the equations above, E is the original sequence of word/line embeddings and W^q, W^k, W^v, b^q, b^k, and b^v are the weights and biases associated with the 1D convolutions used to create each of the new Q, K, and V representations. We scale the dot product QK^T by $\sqrt{d}$ because it has been found to improve performance when the dimension size of the embeddings d becomes very large [141]. Our implementation of self-attention is illustrated in the left two diagrams of Figure 3.2.

Target-Attention

For classification purposes, all sequences regardless of length must be represented as a fixed length vector so that it can be fed into the same softmax classification layer. Our target attention mechanism compares each entry in a sequence $S \in \mathbb{R}^{l \times d}$ with a learned target vector $T \in \mathbb{R}^{1 \times d}$ that represents the information to look for given the current task. T is initialized randomly and then learned through backpropogation during the training progress; entries in S that are more similar to T will be assigned higher importance, and vice versa. The output of the target attention mechanism is a fixed vector $O \in \mathbb{R}^{1 \times d}$ which is a weighted average of the entries in S and captures information from the most critical entries based on the given task. Our target attention mechanism uses similar operations to that of our self-attention mechanism:

$$\text{Target-Attention}(S) = \text{softmax}(\frac{TS^\top}{\sqrt{d}})S \tag{3.3}$$

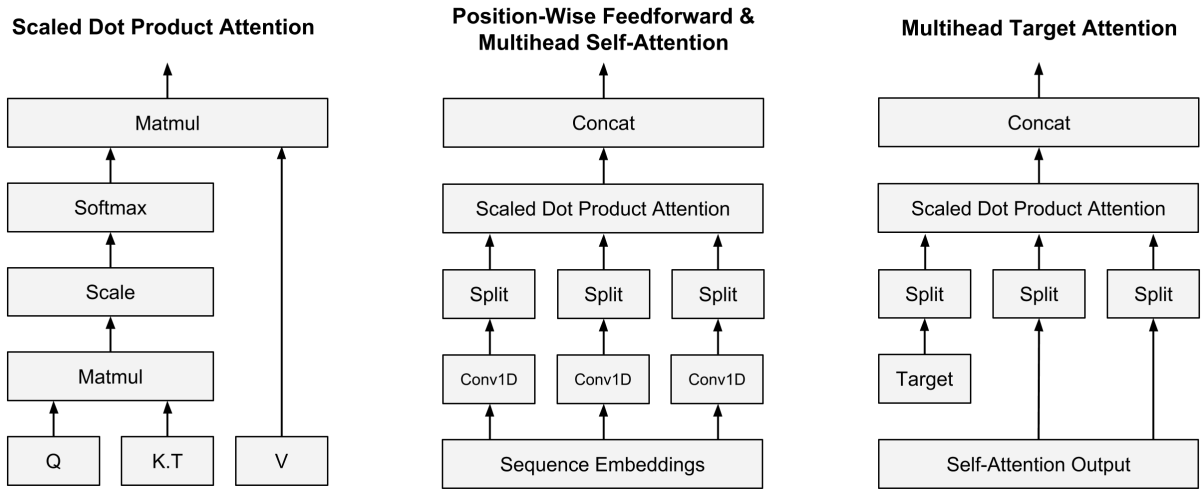


Figure 3.2: Types of attention mechanisms used in the HiSAN.

Our implementation of target-attention is illustrated in the rightmost diagram of Figure 3.2.

Multihead Attention

In both self-attention and target-attention, the attention operations calculate a set of attention weights used to compute a weighted average across the word embeddings; these attention weights are then applied across all dimensions d of the word embeddings. Vaswani et al. found that performance can be improved by using h parallel attention mechanisms, each which attends to a different portion of the embedding dimension d [141]. As a result, different portions of the embedding dimension can be combined using different weights – this captures more complex relations because different semantic concepts are often captured in different subportions of the embedding dimension d [128]. We apply the same principle in our implementations of self-attention and target attention:

$$\begin{aligned} \text{MultiHead Self-Attention}(Q, K, V) &= [head_1, ..., head_h] \\ \text{where } head_i &= \text{Self-Attention}(Q_i, K_i, V_i) \end{aligned} \tag{3.4}$$

$$\begin{aligned} \text{MultiHead Target-Attention}(S) &= [head_1, ..., head_h] \\ \text{where } head_i &= \text{Target-Attention}(S_i) \end{aligned} \tag{3.5}$$

In the equations above, we split each of the Q, K, V, and S embeddings into h sub-embeddings such that $\{Q_i, K_i, V_i, S_i\} \in \mathbb{R}^{l \times d/h}$. Each set of sub-embeddings is then fed into its own self-attention or target attention function. The final output is the concatenation of the outputs $head_i$ from the individual attention functions, resulting in an output sequence $S^{output} \in \mathbb{R}^{l \times d}$ for multihead self-attention and $O^{output} \in \mathbb{R}^{1 \times d}$ for multihead target-attention. A key benefit of this approach is that, unlike other techniques such as adding additional convolution filters, multihead attention increases model's flexibility without increasing the number of trainable model weights relative to regular attention.

Hierarchical Structure

Self-attention finds relationships between the entries in a sequence regardless of how far apart they are in that sequence. This means that no matter where two words appear in a document, self-attention can still identify the same relationship between those two words. To get the most out of this capability, we utilize a hierarchical structure similar to that used in our previous work with HANs [41] – for any given pathology report, we first segment the report into individual lines, then processes each line individually before processing the entire report. This forces the self-attention mechanism to first find local relationships between words in the same line and identify the most important words per line before trying to find global relationships between words across different lines. This allows the HiSAN to better focus on the words and word relationships that matter most to a given classification task.

Our HiSAN uses two hierarchies. For each line in the pathology report, the "word" hierarchy constructs a line embedding representing the content of that line based off the most important word embeddings in that line. Then, the "line" hierarchy constructs a document embedding representing the content of the entire pathology report based off the most important line embeddings generated by the "word" hierarchy. This final document embedding is then used for classification. Each hierarchy in the HiSAN is composed of elementwise feedforward operations to create the Q, K, and V vectors used in self-attention, a multihead self-attention mechanism to find important relationships between the embeddings, and a multi-head target-attention mechanism that constructs the final output embedding representation.

In our previous work on HANs, we found that a hierarchical structure gives a noticeable boost to classification performance on pathology reports. In our experiments, we test the effectiveness of the same hierarchical structure for our HiSAN.

Regularization

Deep learning models are susceptible to overfitting; thus, we utilize dropout to regularize our network. We apply dropout in two ways. First, we dropout 10% of the word and line embeddings that are used in each hierarchy; this type of dropout has been shown to be effective in other NLP applications [109]. Second, we dropout 10% of the outputs from all self- and target-attention mechanisms (within each multihead attention) as follows:

$$\text{Self-Attention}(Q, K, V) = \text{dropout}\left(\text{softmax}(\frac{QK^T}{\sqrt{d}})\right)V \tag{3.6}$$

$$\text{Target-Attention}(S) = \text{dropout}\left(\text{softmax}(\frac{TS^\top}{\sqrt{d}})\right)S \tag{3.7}$$

Dropping out random words in the normalized similarity matrix generated after the softmax function reduces overfitting by preventing the attention mechanisms from always learning the same relationships between different words, thereby forcing the attention mechanisms to explore new potential word relationships.

We found that utilizing these two forms of dropout improved the performance of our HiSAN across all five classification tasks.

3.5 Results

Our experiments consist of comparing the effectiveness of the HiSAN in classifying cancer pathology reports in five key data elements – site, laterality, behavior, histology, and grade – against other popular text classification methods. We compare the methods in accuracy, macro F-score, and speed.

3.5.1 Baselines, Hyperparameters, and Setup Details

We compare the performance of our HiSAN against both traditional ML and deep learning algorithms commonly used on medical text data as well as against other deep learning architectures that have been previously applied on medical text classification. The hyperparameters of all baseline models and the HiSAN were tuned on a separate 2-fold cross validation setup on the reports from 2004-2015, which includes the train and validation data but excludes the test data used to report our final results; this was done to expand the dataset size used for hyperparameter tuning in order to find more robust hyperparameters.

For non-deep learning approaches, we compare against Naive Bayes (NB) and logistic regression (LR) using TF-IDF unigrams and bigrams as features; due to our large dataset of 375K documents and the large feature space of 780K unique unigrams and bigrams, other approaches such as support vector machines and random forests were prohibitively expensive in terms of memory and/or time. We tuned the hyperparameters for LR on our validation set – we use L1 regularization with a penalty strength of 1.0.

For deep learning approaches, we compare against two popular text classification architectures that have previously been benchmarked on cancer pathology reports - a word-level convolutional neural network (CNN) and a hierarchical attention network (HAN). In our previous with work HANs [41], we found than RNNs without the hierarchical structure not only perform worse than HANs in accuracy and macro F-score for pathology report classification, but also take longer to train. Therefore, these have been excluded from our study.

The word-level CNN is based off the architecture developed by Kim in 2014 for sentence classification [68]. In 2017, Qiu et al. showed that a similar architecture outperforms non-deep learning approaches when classifying site and grade from cancer pathology reports [116]. For our text CNN, we use the similar hyperparameters as those specified by Qiu – three parallel convolutional filters that process 3, 4, and 5 words at a time, each with 300 filters, followed by a maxpooling operation across the entire document.

The HAN was originally developed by Yang et al. in 2016 for sentiment analysis [152]. Previously in 2018, we showed that the HAN beat out both the text CNN and non-deep

learning approaches in identifying site and grade in cancer pathology reports [41] – we consider the HAN to be the current state-of-the-art in identifying key data elements in cancer pathology reports. For our HAN, we use the same hyperparameters as those used in our previous paper – each hierarchy is composed of a bi-directional GRU with 200 units and an attention mechanism with a hidden layer of 300 neurons.

In addition to the CNN and HAN, we also compare against a non-hierarchical version of the HiSAN, which we call the self-attention network (SAN), to show that use of the hierarchical structure indeed improves performance. For both our SAN and HiSAN, we use a dimension size of 512 for d and 8 heads for h in our position-wise feedforward, multihead self-attention, and multihead target-attention operations for all tasks except behavior, which uses 256 for d.

For our deep learning approaches, we generate word embeddings for each word in our vocabulary using the popular Word2Vec algorithm [94]. We train our Word2Vec embeddings on our entire corpus of pathology reports using a dimension size of 300, a window size of 5 words, and the continuous-bag-of-words algorithm. These word embeddings are set as learnable parameters during training so that each word embedding can be further tuned to maximize performance in our classification tasks.

All non-deep learning approaches are implemented using the `scikit-learn` package in Python; these are each fitted on the train set and evaluated on the test set. All deep learning models are implemented using the `Tensorflow` package in Python; these are each trained on the train set using a batch size of 64 and the Adam optimizer [69] with learning rate 1E-4, beta1 0.9, and beta2 0.99. After every epoch, accuracy is evaluated on the validation set; training stops when validation accuracy fails to improve for 5 consecutive epochs. For each deep learning model, we save the model parameters following the epoch with the highest validation accuracy and use those parameters to evaluate on the test set.

3.5.2 Evaluation Metrics

Many of these classification tasks have extreme class imbalance – the pathology reports are distributed very unevenly between the possible classes. In particular, for histology, there are over 500 possible classes yet many of these histologies have fewer than 100 instances

per class (see Appendix B.1 for histograms of instances per class). Thus, we evaluate the performance of each classification approach using two distinct metrics – overall accuracy and macro F-score. We note that in classification tasks such as ours in which each report is assigned to exactly one class, accuracy is the same as micro F-score.

In each task, overall accuracy simply measures the percentage of reports in the test set classified correctly. This metric does not disproportionately penalize a classifier for performing poorly on the less populated classes.

Macro F-score, on the other hand, is more heavily influenced by how well the classifier performs on the less prevalent classes. Intuitively, macro F-score measures classifier performance within each class, and then averages this performance across all classes; therefore, performing well in a rare class is equally important to performing well in a common class. Whereas accuracy measures how well a classifier performs overall, macro F-score better captures how well a classifier can identify cases that it does not see often, which is highly important in real-world settings. The mathematical implementation for macro F-score is described below:

$$\begin{aligned}
\text{Precision}_c &= \frac{\text{True Positives}_c}{\text{True Positives}_c + \text{False Positives}_c} \\
\text{Recall}_c &= \frac{\text{True Positives}_c}{\text{True Positives}_c + \text{False Negatives}_c} \\
\text{F1 Score}_c &= \frac{2 \times \text{Precision}_c \times \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} \\
\text{Macro F1 Score} &= \frac{1}{n} \sum_{c=i}^{n} \text{F1 Score}_c
\end{aligned} \tag{3.8}$$

where n is the total number of possible classes within a given classification task and c is a specific class.

For both accuracy and macro F-score, we establish 95% confidence intervals for each metric by bootstrapping samples from the test set – our procedure is described in Appendix B.2. We use these confidence intervals to establish statistical significance between the difference in performance across the different approaches. We also measure training and inferences times for all classifiers – all models are timed on the same machine using four Intel Xeon E5-2695 v4 CPUs and one Tesla P100 GPU.

3.5.3 Experimental Results

The HiSAN achieves the highest accuracy and macro F-score across all five tasks, even compared to the HAN (Table 3.1). The performance of the HiSAN is especially noticeable for macro F-scores, which indicates that the HiSAN approach is much better at accurately classifying cases with very few examples. This is important for practical use because certain cancer types or histologies may be relatively rare but still need to be identified accurately.

To further verify the statistical significance of the difference in performance between classifiers, we perform a McNemar's test [30] between the test set predictions of every pair of classifiers. Our results are shown in Appendix B.4 and confirm that the HiSAN's performance is significantly different than that of all other classifiers across all classification tasks.

We show the F-score breakdown by class for each classifier on each classification task in Appendix B.3. The HiSAN achieves the best F-score across the most classes, especially for the most prevalent classes; furthermore, when the HiSAN is not the top performer, it is still one of the top performers in the group. In the severely underrepresented classes (<1% of total samples), there is less consistency on which classifier performs best. We also note that some classes may be easier to identify than others – for example, in the site task, all classifiers are far better at identifying C60 (penis) than C48 (retroperitoneum and peritoneum) or C57 (other and unspecified female organs) even though they each have a similar number of training samples (0.13% - 0.18% of total samples). A manual examination revealed that the language for certain cancers is very distinct from all other cancers, making them easier to identify, while certain groups of cancers have significant language overlap, making them harder to distinguish; this is examined further in Sections 3.6.1 and 3.6.2.

In Table 3.2, we rerun our experiments on the site and histology tasks using a reduced dataset in which we remove all reports from the 15 most prevalent cancer sites. We keep the same train/validation/test splits as the original dataset; this results in 38580 train samples, 9714 validation samples, and 11600 test sammples. Using this reduced dataset, the difference in performance between the HiSAN and the other classifiers is much clearer – this shows that when fewer samples are available for training, the HiSAN is better able to handle class imbalance.

Table 3.1: Top: test set accuracy and macro F-score (with 95% confidence intervals) on each classification task. Bottom: train and inference times (in seconds) for each classification method; train times for deep learning approaches are displayed as time per epoch × number of epochs to converge.

	Site		Laterality		Behavior		Histology		Grade	
	Acc	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**
NB	66.8 (66.4, 66.9)	11.7 (11.7, 11.9)	80.5 (80.2, 80.7)	32.5 (32.4, 32.7)	91.1 (91.0, 91.3)	26.7 (26.4, 27.0)	46.3 (46.1, 46.7)	1.54 (1.48, 1.56)	51.5 (51.00, 51.6)	26.8 (26.5, 26.8)
LR	88.1 (88.1, 88.5)	56.6 (56.2, 57.9)	88.4 (88.4, 88.7)	43.3 (43.0, 45.1)	96.1 (95.9, 96.2)	82.1 (78.1, 82.8)	72.8 (72.6, 73.2)	24.4 (24.0, 25.4)	66.6 (66.5, 67.1)	60.7 (59.4, 63.8)
CNN	89.4 (89.3, 89.6)	56.4 (55.6, 57.1)	89.1 (88.9, 89.2)	47.9 (46.9, 48.9)	96.4 (96.3, 96.6)	75.0 (72.6, 77.3)	75.4 (75.2, 75.7)	23.5 (23.4, 24.9)	70.0 (69.7, 70.3)	68.7 (66.4, 70.8)
HAN	89.9 (89.7, 90.0)	61.2 (60.4, 61.9)	89.0 (88.8, 89.2)	47.7 (46.6, 48.7)	96.5 (96.4, 96.6)	80.3 (77.8, 82.7)	75.7 (75.4, 75.9)	26.8 (26.8, 28.3)	71.1 (70.9, 71.4)	74.0 (72.8, 75.0)
SAN	89.1 (89.0, 89.3)	59.6 (58.8, 60.9)	88.9 (88.7, 89.0)	49.6 (48.6, 50.6)	96.4 (96.3, 96.5)	82.2 (79.8, 84.3)	74.1 (73.8, 74.3)	26.1 (26.0, 27.5)	67.6 (67.3, 67.9)	69.2 (67.4, 70.5)
HiSAN split/10 tok	90.2 (90.0, 90.4)	62.7 (61.7, 63.4)	89.3 (89.2, 89.5)	48.8 (48.8, 50.8)	96.7 (96.6, 96.8)	82.9 (80.4, 85.1)	76.2 (75.9, 76.4)	29.9 (29.7, 31.5)	71.3 (71.1, 71.6)	74.1 (72.7, 75.1)
HiSAN split/line	**90.4** (90.2, 90.5)	**63.4** (62.3, 64.1)	**89.4** (89.2, 89.5)	**50.0** (49.1, 51.0)	**96.7** (96.6, 96.8)	**84.0** (81.6, 86.1)	**76.2** (76.0, 76.5)	**30.2** (30.1, 31.8)	**71.6** (71.3, 71.9)	**74.3** (73.0, 75.3)

	Train Time (s)					Avg Pred Time (s)
	Site	**Laterality**	**Behavior**	**Histology**	**Grade**	**For 64 Docs**
NB	1.415E+1	3.29E+0	2.45E+0	7.94E+1	3.38E+0	4.61E-3
LR	2.01E+3	2.82E+2	1.56E+2	1.33E+4	4.32E+2	3.59E-3
CNN	1.06E+3 × 13 ep	1.06E+3 × 8 ep	1.05E+3 × 9 ep	1.06E+3 × 14 ep	1.06E+3 × 13 ep	1.01E-1
HAN	5.11E+4 × 5 ep	5.04E+4 × 4 ep	5.00E+4 × 5 ep	5.08E+4 × 7 ep	5.05E+4 × 7 ep	4.69E+0
SAN	1.28E+3 × 20 ep	1.27E+3 × 11 ep	1.27E+3 × 15 ep	1.28E+3 × 19 ep	1.28E+3 × 18 ep	1.52E-1
HiSAN split/10 tok	2.19E+3 × 8 ep	2.18E+3 × 5 ep	2.17E+3 × 7 epochs	2.19E+3 × 10 ep	2.18E+3 × 12 ep	2.22E-1
HiSAN split/line	3.16E+3 × 9 ep	3.14E+3 × 6 ep	3.13E+3 × 7 ep	3.17E+3 × 10 ep	3.14E+3 × 10 ep	4.52E-1

Table 3.2: Test set accuracy and macro F-score (with 95% confidence intervals) on the site and histology tasks using a reduced dataset in which cases from the 15 most prevalent cancer sites are removed.

	Site		Histology	
	Acc	**Macro**	**Acc**	**Macro**
NB	42.02 (40.35, 41.87)	15.19 (14.61, 15.23)	32.21 (31.92, 33.30)	0.74 (0.69, 0.80)
LR	72.24 (70.94, 72.30)	57.82 (55.68, 58.12)	65.69 (64.73, 66.19)	25.26 (23.73, 26.00)
CNN	73.10 (72.37, 73.72)	56.63 (55.47, 57.61)	67.53 (66.84, 68.25)	20.69 (20.41, 22.37)
HAN	72.81 (72.12, 73.50)	57.30 (56.20, 58.17)	68.90 (68.23, 69.61)	22.93 (22.63, 24.81)
HiSAN split/line	**75.16** (74.49, 75.86)	**62.52** (61.19, 63.51)	**70.41** (69.70, 71.11)	**27.71** (27.31, 29.94)

The HiSAN performs better than the SAN in both accuracy and macro F-score across all tasks, which suggests that utilizing a hierarchical structure that breaks down a long document into smaller chunks helps the model better locate text segments critical for classification, particularly in the more difficult tasks like histology and grade. This is consistent with our findings from our previous work with HANs – neural architectures specifically designed to process sequential data like RNNs and self-attention can locate important linguistic relationships over long text segments, but when these sequences get too long these architectures are unable to locate or retain critical information. By breaking down long documents like pathology reports into shorter segments like individual lines, we can take advantage of the benefits of RNNs or self-attention with far less information loss. We test two version of hierarchical splitting within the HiSAN, one which splits pathology reports into chunks of 10 tokens each, and one which splits pathology reports by line breaks (as described in our data cleaning section). We see that while both versions significantly outperform the SAN, splitting based on the line breaks performs slightly better – this makes sense because in most cases, the line breaks reflect how the original author organized information within the pathology report. For the remainder of this paper, references to the HiSAN will refer to the HiSAN that splits by lines.

In terms of timing, the HiSAN takes roughly 2-3x longer to train and 5x longer for inference compared to the CNN (Table 3.1, Figure 3.3). Relative to the HAN, the HiSAN completes training before the HAN completes a single epoch, and the HiSAN makes predictions over 10x faster than the HAN.

We note that our implementation of self-attention does not utilize positional embeddings. On its own, self-attention does not distinguish the order of words in an input sequence; therefore, many implementations of self-attention-based architectures utilize positional embeddings to capture information about word order [43, 31]. However, we found that adding positional embeddings to the HiSAN did not consistently affect performance on our tasks (Appendix B.7).

Furthermore, we found that adding additional architectural components from the HCAN [40] designed to capture more complex linguistic relationships reduce the performance of the HiSAN on pathology report classification (Appendix B.7). We expect that this is because

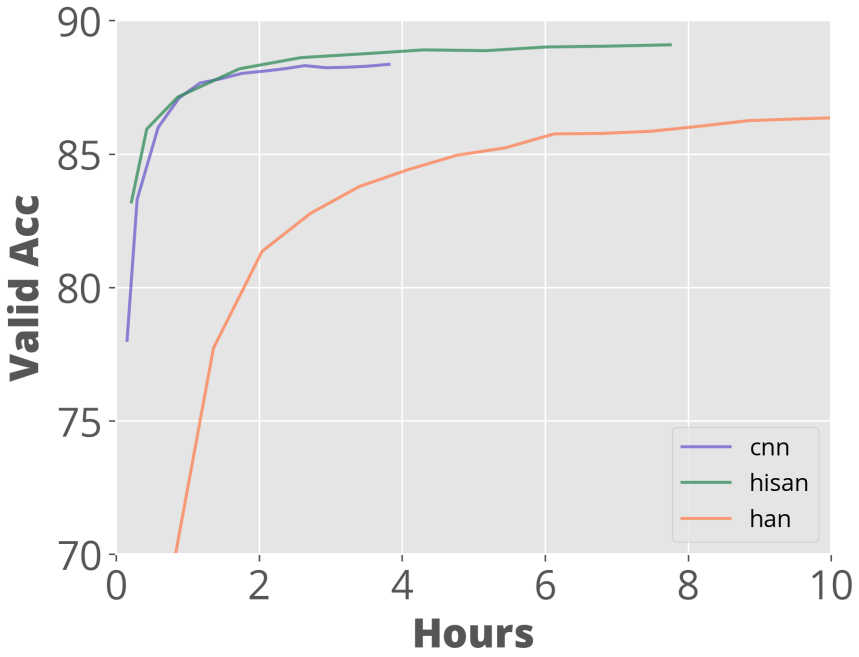


Figure 3.3: Validation accuracy vs time on site classification task.

the percentage of relevant words in cancer pathology reports is very low for our given tasks – only a few clinical concepts are useful for identifying a particular label. Therefore, compared to a task like sentiment analysis on short Yelp Reviews [40], word order and more complex linguistic features captured by additional architectural components such as parallel attention mechanisms are less important for our classification tasks.

In our main set of experiments, we replace rare words (appearing fewer than five times in the corpus) with an "unknown_word" token. In Appendix B.8, we also show the performance of the HiSAN in which all possible words are included in the word embedding vocabulary matrix. Including all words gives small boost in performance in the histology task – we expect that this is because histology has the highest number of possible classes and highest class imbalance, so rare words may be useful for identifying rare classes. However, including all vocabulary decreases performance on all other tasks.

3.6 Discussion

3.6.1 Error Analysis

As described in our Dataset Details section, it is important to note that our ground truth labels are at the tumor-level rather than at the report-level – the label for each report is assigned based off all available data associated with the tumor ID associated with that report. Thus, it is possible that the tumor-level label may not be appropriate for a specific report if that report is for a secondary or metastatic site.

To better understand how and why the HiSAN was misclassifying pathology reports, we manually analyzed 200 randomly selected pathology reports that were misclassified by the HiSAN. We focused on the site task because site is relatively easy for non-experts to identify compared to the other classification tasks. Based off our analysis of this subset, we identified four main categories of misclassified reports.

In the first category, the report is misclassified because there does not appear to be any information associated with the ground truth site in the report; therefore, the ground truth site may be difficult or impossible to identify even by an expert. This is most likely because the report is an addendum or biopsy of a secondary or metastatic site. Out of 200 misclassified reports examined, 58 reports (29%) fell into this category.

In the second category, the report mentions two or more (usually metastatic) sites. For these reports, the HiSAN may incorrectly predict a secondary site instead of the primary site. Out of 200 misclassified reports examined, 21 reports (10.5%) fell into this category.

In the third category, the predicted site is close to ground truth site (i.e., within the same organ system or a neighboring organ), and the predicted site is often directly mentioned in the pathology report. For example, the ground truth site may be rectosigmoid junction and the HiSAN may predict colon, or the ground truth site may be cervix and the HiSAN may predict uterus. Out of 200 misclassified reports examined, 104 reports (52%) fell into this category. From these 104 reports, we attempted to manually predict the site without knowing the ground truth site or the HiSAN's predicted site. We predicted the ground truth site in 29% of the reports, agreed with the HiSAN's prediction in 60% of the reports, and chose a different incorrect site in the remaining 11% of the reports.

The fourth and last category involves ill-defined sites (C76), unknown sites (C80), or a general catch-all site for a particular organ system (e.g., C57 unspecified female genital organs). Either the ground truth site is one of these ill-defined sites or the HiSAN mispredicts one of these ill-defined sites, despite the report mentioning specific organs or cancer sites. Out of 200 misclassified reports examined, 17 reports (8.5%) fell into this category.

In addition, we generated and analyzed the confusion matrix from the HiSAN for each of the five tasks (available in Appendix B.3). In the site task, most misclassifications were between sites associated with neighboring or highly related organs; four commonly confused groups of sites were (1) between C42 hematopoietic and reticuloendothelial systems, C44 skin, and C77 lymph nodes, (2) between C51 vulva, C52 vagina, C53 cervix, and C54 uterus, (3) between C64 kidney, C65 renal pelvis, C66 ureter, and C67 bladder, and (4) between C18 colon, C19 rectosigmoid junction, C20 rectum, and C21 anus. This is consistent with our manual analysis of the misclassified reports. Unsurprisingly, the classes with the lowest accuracy tended to be the classes with the least number of occurrences in the training set; this is similar to how overall accuracy on the site task dropped for all classifiers in our experiment in which we removed reports from the 15 most prevalent cancer sites.

Our analysis shows that, for the site task, the misclassifications of the HiSAN (and other ML models) may be attributable to other factors, such as data mislabelling. To verify this, we tested the performance of all the trained models on a reduced test set of 9481 pathology reports. In this reduced test set, all reports belong to a Tumor ID that is associated with only a single pathology report, thereby eliminating any addendums and supplementary reports. The results of these experiments are available in Appendix B.5. Compared to our results on the standard test set, we saw around 20-25% relative improvement in accuracy for the HiSAN across all tasks (measured by decrease in classification error) and around 4-24% relative improvement in macro F-score. Interestingly, the performance of NB, LR, CNN, and HAN slightly decreased or saw no significant improvement on the site, histology, and sometimes other tasks. We expect that this is because our reduced test set contains more minority classes in which these classifiers are less accurate (see Appendix B.3). Taken together, the benefit of using the HiSAN compared to the other classifiers is more apparent across all tasks on this reduced test set.

3.6.2 Visualizing Document Embeddings

The penultimate layer of the HiSAN encodes each pathology report into a vector representation of fixed size – this "document embedding" represents the critical content within that report for a given classification task. In left side of Figure 3.4, we visualize the test set document embeddings on the site task colored by their ground truth cancer type, reduced into 2D-space via t-statistic stochastic neighborhood embedding (t-SNE) [140]. This allows us to better understand how the HiSAN encodes pathology reports associated with different types of cancers and also shows us the overlap of keywords in pathology reports associated with different types of cancers. We see that the HiSAN neatly separates the different cancer types into separate clusters.

This type of visualization is useful for analyzing misclassified pathology reports. The right side of Figure 3.4 shows only the document embeddings of the misclassified reports within the same test set. Using the same random sample of 200 misclassified reports from our error analysis, we examined the text of the report, the predicted and true labels, and the location of the document embedding in our visualization. We found that misclassified reports that are still located in a cluster belonging to the same organ system usually indicate that the HiSAN mispredicted a neighboring organ, while misclassified reports located in clusters associated with completely different organ systems usually indicate that the report mentions multiple sites or is about a secondary site rather than the primary site.

Our visualization shows that within the site task, certain organ systems, such as digestive and urinary tract, tend to have higher rates of misclassification. Based off our manual examination, we found that there tends to be overlap in the language and terminology used for cancer sites belonging to certain organ systems (e.g., ureter cancer may also mention bladder), and therefore confusion is more likely.

3.6.3 Visualizing Word Importance

A major benefit of the HiSAN is its transparency in how it makes decisions. Each hierarchy of the HiSAN is composed of attention mechanisms, which directly assign importance weights to the words/lines it is analyzing. We can therefore analyze the attention weights at both

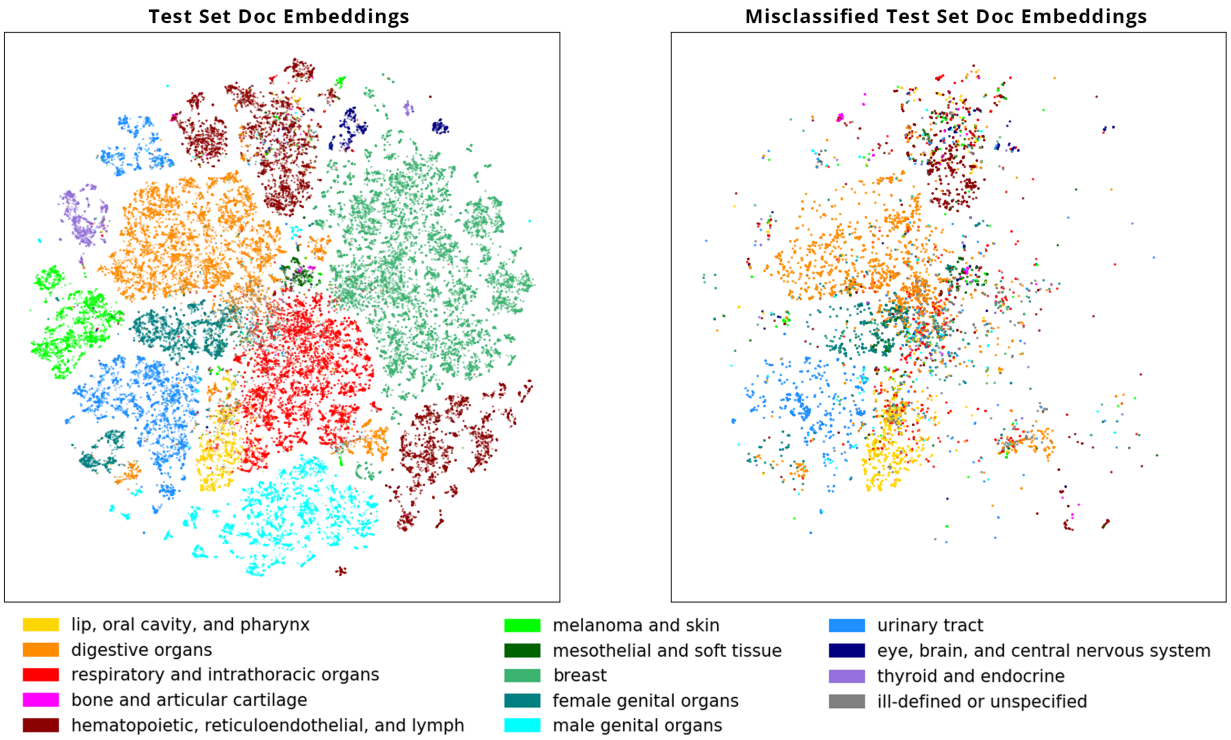


Figure 3.4: The HiSAN encodes each document into an fixed-size vector representation (aka "document embedding") that we can visualize in 2D using t-SNE. The left figure shows the document embeddings for all entries in the test set and right figure shows the document embeddings for all misclassified entries in the test set for the site task.

the self-attention and target-attention levels to see first how the HiSAN finds relationships between different words/lines and then how the HiSAN chooses which words/lines are most relevant to a given task.

The top portion of Figure 3.5 show an example pathology report annotated with the words that the HiSAN used to make its decision for the site task. We identified important words by following the trace of the attention weights across all the attention mechanisms in the HiSAN; we use the aggregate attention weights across all attention heads in the HiSAN. This can be a highly useful tool for human annotators when manually annotating a pathology report – because the HiSAN highlights the keywords in each report associated with the target classification task, the human can focus on those sections of the report to reduce the amount of time required to come to a decision. We also used these annotations to examine how and why the HiSAN fails in misclassified pathology reports, which were described above in our error analysis section. For example, if the HiSAN identifies keywords belonging to two entirely distinct cancer sites in the same report, it may suggests that metastasis is present or that the report is about a secondary site.

The attention weights associated with the HiSAN's self-attention mechanism show how the HiSAN finds relationships between different words and lines. The bottom portion of Figure 3.5 shows the self-attention weights associated with the word "breast" within a particular line of a report; we use the aggregate self-attention weights across all attention heads in the HiSAN. We see that the HiSAN identifies that both "ductal carcinoma" and "comedo necrosis" are referring to "breast"; a CNN-based model would have trouble capturing these relationships because CNNs can only identify relationships within a fixed window size, typically five words or less. This suggests that one reason the HiSAN outperforms CNNs is that it can capture more long distance linguistic relationships important for classification.

Finally, the HiSAN utilizes multihead attention for all its attention mechanisms. We show in Appendix B.6 how different attention heads to pay attention to different relationships between words/lines. This is similar to ensembling with parallel networks, with each network analyzing a different portion of the embedding dimension of the words in a pathology report. The end result is a minor performance boost compared to using the same attention weights

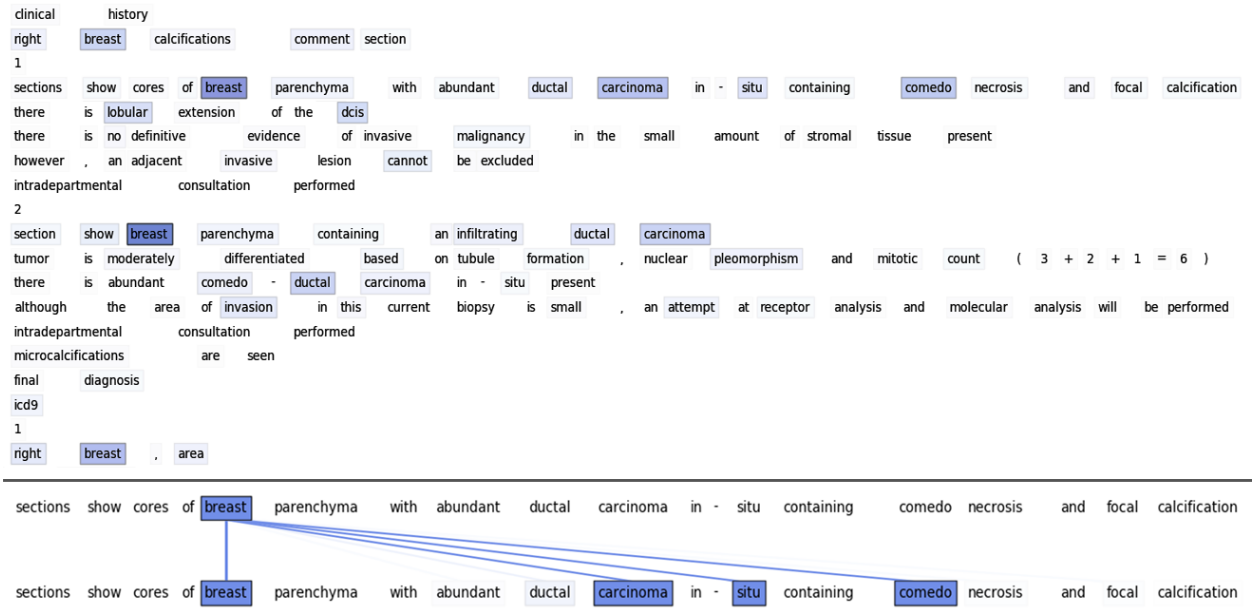


Figure 3.5: A section of a pathology report on breast cancer. In the top figure, we can follow the attention weights across all the attention layers in the HiSAN to visualize the words used to make the final classification decision. In the bottom figure, we can examine the self-attention weights associated with a particular word (in this case, the word "breast") to visualize how the HiSAN finds relationships between individual words. We note that for both figures, we show the aggregate attention weights across all attention heads in the HiSAN.

across the entire embedding dimension (see Appendix B.6 for a comparison of HiSAN performance using different numbers of attention heads). For annotation purposes, using the aggregate attention weights across all attention heads (as we did in Figure 3.5) provides the most comprehensive visualization of all words and word relationships relevant to a given classification task, as it includes the different relationships captured within each individual attention head.

3.7 Conclusion

In this paper, we presented the Hierarchical Self-Attention Network (HiSAN), a deep learning architecture designed for information retrieval from pathology reports. We demonstrated its superior performance relative to other traditional ML and deep learning text classification techniques across five classification tasks – site, laterality, behavior, histology, and grade – using a corpus of 374,899 pathology reports. The HiSAN achieved the best accuracy and macro-F score across all five tasks; its performance relative to the other classifiers was especially strong on the less prevalent classes. Furthermore, the HiSAN trains more than an order of magnitude faster than the previous state-of-the-art, HANs, and its attention-based architecture enables easy visualization of its decision-making process for any given pathology report.

One finding of our study was that the HiSAN (and other classifiers) have more difficulty accurately classifying reports belonging to tumor IDs associated with multiple reports. This is because our ground truth labels were at the tumor level rather than report level. We found the performance of the HiSAN to improve significantly by limiting our test set to reports belonging to tumor IDs associated with only a single report. Therefore in practice, the performance of the HiSAN may be improved by restricting the set of pathology reports it is applied to (e.g. excluding metastatic at diagnosis and excluding reports that are dated long after diagnosis). For future work, we plan to examine methods that account for contextual information from all reports associated with the same unique tumor ID – for example, an additional report-level hierarchy can be used to capture report-level context and identify which reports are most relevant for identifying the ground truth labels for each tumor ID.

Although our HiSAN achieves higher accuracy and macro F-score across all tasks compared to the previous state-of-the-art and all other baselines, there is still room for improvement on the difficult tasks, such as histology and grade, in order to bring it to a level in which it may be deployed in place of a human expert. For these more difficult tasks, uncertainty quantification methods may be utilized so that predictions with high confidence can be used without human review, thus reducing the burden on human annotators. We are currently exploring this aspect in partnership with NCIs SEER program. The code for the HiSAN is available online at github.com/iamshang1/Projects/tree/master/Papers/HiSAN.

Chapter 4

Using Case-Level Context to Classify Cancer Pathology Reports

A version of this chapter is currently being submitted to academic journals.

Shang Gao designed, implemented, and ran the experiments. Shang Gao, Mohammed Alawad, Jacob Hinkle, Noah Schaefferkoetter, Hong-Jun Yoon, and Blair Christian developed the text preprocessing and experimental pipeline. Paul Fearn, Linda Coyle, Xiao-Cheng Wu and Linda Coyle oversaw and organized the preparation of the pathology reports. Shang Gao wrote the manuscript with support from Arvind Ramanthan and Georgia Tourassi. All authors read and approved the manuscript.

This chapter is identical to the version being submitted to journals.

4.1 Abstract

Individual EHRs and clinical reports are often part of a larger sequence – for example, a single patient may generate multiple reports over the trajectory of a disease. In applications such as cancer pathology reports, it is necessary not only to extract information from individual reports, but also to capture aggregate information regarding the entire cancer case based off case-level context from other reports in the sequence. In this paper, we introduce a simple modular add-on for capturing case-level context that is designed to be compatible with most existing deep learning architectures for text classification on individual reports. We test our approach on a corpus of 431,433 cancer pathology reports, and we show that incorporating case-level context significantly boosts classification accuracy across six classification tasks – site, subsite, laterality, histology, behavior, and grade. We expect that with minimal modifications, our add-on can be applied towards a wide range of other EHR-based tasks.

4.2 Introduction

Electronic health records (EHRs) are a prevalent and detailed source of health data – according to the Office of the National Coordinator for Health Information Technology, as of 2017, 86% of office-based physicians store health records electronically[1]. These EHRs record detailed information from all the clinicians involved in a patient's care – this can include

[1]https://dashboard.healthit.gov/quickstats

demographics, progress notes, medications, vital signs, past medical history, immunizations, laboratory tests and results, radiology reports, and more [92]. As a result, EHRs are an important tool for public health surveillance and for monitoring communicable and chronic diseases [10].

In recent years, deep learning has played an important role in automating information extraction from EHRs [131]. Whereas traditional machine learning approaches such as logistic regression and support vector machines require human experts to manually engineer input features, deep learning architectures are able to learn salient features directly from raw data. Deep learning models often outperform traditional machine learning techniques because deep learning models are capable of recognizing nuanced and complex patterns that are difficult to capture manually – this is especially true for raw data in the form of unstructured text and natural language [106]. In the domain of EHRs, deep learning approaches have been used for missing data imputation [16], patient phenotyping [108], cohort identification [44], clinical decision support [126], and much more.

One notable property of EHRs is that they often come in a sequence - a single patient or case may generate multiple reports over time. Within the same sequence, EHRs are generally related to each other in some manner; for example, the diagnosis of a disease in one EHR may prompt additional tests for that disease in following EHRs, and later EHRs may document the treatment or progression of the disease. In some applications, for any given clinical report, it is helpful or necessary to extract aggregate information using other reports in the sequence. An important example is cancer pathology reports – individual cancer pathology reports are often tagged with aggregate labels that describe the cancer case as a whole, and these aggregate labels require collective analysis of all pathology reports belonging to a given cancer case. Utilizing contextual information from multiple EHRs in a sequence may improve performance across a variety of medical tasks [161, 60, 133].

There are many existing studies on how deep learning can be applied on unstructured text and natural language from EHRs [147]; however, to our knowledge, no studies have been done on how deep learning can utilize case-level context to capture aggregate information from a sequence of unstructured clinical text. In this paper, we present a simple modular add-on for capturing and utilizing case-level context that is designed to be compatible with most

existing deep learning architectures for classifying individual documents. We test our add-on with two existing deep learning architectures – word-level convolutional neural networks (CNNs)[68], which are widely used across many EHR-based applications [102, 7, 15, 49], and hierarchical self-attention networks (HiSANs)[], the current state-of-the-art in cancer pathology report classification. We show that our add-on improves the effectiveness of both networks in classifying six key data elements – site, subsite, laterality, behavior, histology, and grade – from a corpus of approximately 430K cancer pathology reports. We expect that with minimal modifications, our add-on may improve performance across a wide range of other EHR- and clinical text-based tasks.

4.3 Background

Because of the inherent temporal aspect of EHRs, existing work has explored how to predict clinical events, phenotype patients, and perform other medical tasks based off structured time-series data extracted from a patient's EHRs. For example, Cheng et. al. used a CNN-based architecture on sequentially-ordered medical events (e.g., ICD-9 codes) extracted from EHRs to predict the onset of Congestive Heart Failure and Chronic Obstructive Pulmonary Disease [18], and Lipton et. al. used a recurrent neural network (RNN) based architecture on a time-series of clinical measurements (e.g., blood pressure and heart rate) extracted from EHRs to diagnose diseases [84]. To our knowledge, existing research that utilizes temporal analysis of EHRs does not use raw natural language as input; rather, they utilize pre-extracted features from EHRs, such as diagnosis codes, medication codes, and procedure codes. As a result, relevant information in the form of natural language, such as those from clinical notes, is not captured by these approaches.

Existing work has also explored how to extract useful information from natural language in EHRs without incorporating any case-level context. For example, Mullenbach et. al. use a CNN-based architecture to extract medical event codes from individual clinical notes [96], and Jagannatha and Yu perform the same task utilizing an RNN-based architecture [58]. Because these approaches process each document independently from any others, any useful relationships between EHRs belonging to the same patient or case are not captured.

There is a small body of existing research on incorporating sentence-level context in the realm of general NLP tasks outside of the EHR domain. Dernencourt and Lee examined how sentence-level context could be used to improve classification of short text sequences in day-to-day dialog [75], and Jin and Szolovits used a similar approach to examine how sentence-level context could improve classification of individual sentences in biomedical and scientific abstracts [61]. The results from these works show that taking advantage of contextual information outside of the target sentence boosts the performance of certain tasks. We propose building upon these works and extending them to the domain of natural language in EHRs – we expect that performance in information extraction from unstructured medical text can be improved by accounting for contextual information from related text, such as those from other EHRs belonging to the same patient.

Our work in this paper builds upon our previous work with CNNs [116, 118, 3] and HiSANs [] on classifying cancer pathology reports. These methods are designed to extract key data elements such as site, histology, and grade from individual cancer pathology reports independent from any other (possibly related) reports. In this work, we show that by adding a modular component for capturing case-level context from other cancer pathology reports from the same tumor case, we can significantly improve the performance of these methods across six classification tasks – site, subsite, laterality, histology, behavior, and grade.

4.4 Materials and Methods

4.4.1 Problem Description

Suppose we have a sequence of n text-based EHRs (e.g., clinical notes) $d_0, d_1, ..., d_n$ which are ordered by the date the report was created. All reports in the sequence are related to each other – for example, all reports belong to the same patient or case. Each report is associated with a label y_i, where y_i is the label for the ith report. The task is to predict the labels y_i for each document d_i in the sequence.

In the baseline case, which has been explored in previous research, a machine learning or deep learning model predicts the label y_i for d_i independently from any other reports

in the sequence. In other words, y_i = Predict(d_i). In this paper, we explore methods to incorporate contextual information from all reports in the sequence, such that y_i = ContextAwarePredict($d_i|d_0, ..., d_n$).

To simulate applications in the real world, we apply restrictions based off two different scenarios. In the first scenario, when processing a report d_i, all other reports in the sequence are available; the predictive model can utilize contextual information from other reports that came both before and after the target report. This first scenario represents offline applications using historical data where for any given patient/case, all EHRs for that patient/case are available.

In the second scenario, when processing a report d_i, only reports that came before d_i are available; the predictive model can only utilize contextual information from reports that came before the target report such that y_i = ContextAwarePredict($d_i|d_0, ..., d_{i-1}$). This second scenario represents online applications where EHRs must be immediately processed as they arrive and information from future reports does not yet exist.

4.4.2 Capturing Case-Level Context

We explore five different methods for incorporating case-level context when extracting information from text-based EHRs – concatenation, RNNs, RNNs with linear-chain conditional random field (CRF), self-attention, and self-attention with linear-chain CRF. These are described in greater detail in the following sections. Figure 4.1 illustrates the baseline case (without incorporating case-level context) and each of the five methods.

Concatenation

The most simple and naive way to incorporate case-level context is to concatenate all reports belonging to the same patient/case, as shown in Equation 4.1. Because the model has access to information from all reports in the sequence, it can utilize information from other reports for decision making on any given report.

$$y_i = \text{Predict}([d_0, ..., d_i, ..., d_n]) \tag{4.1}$$

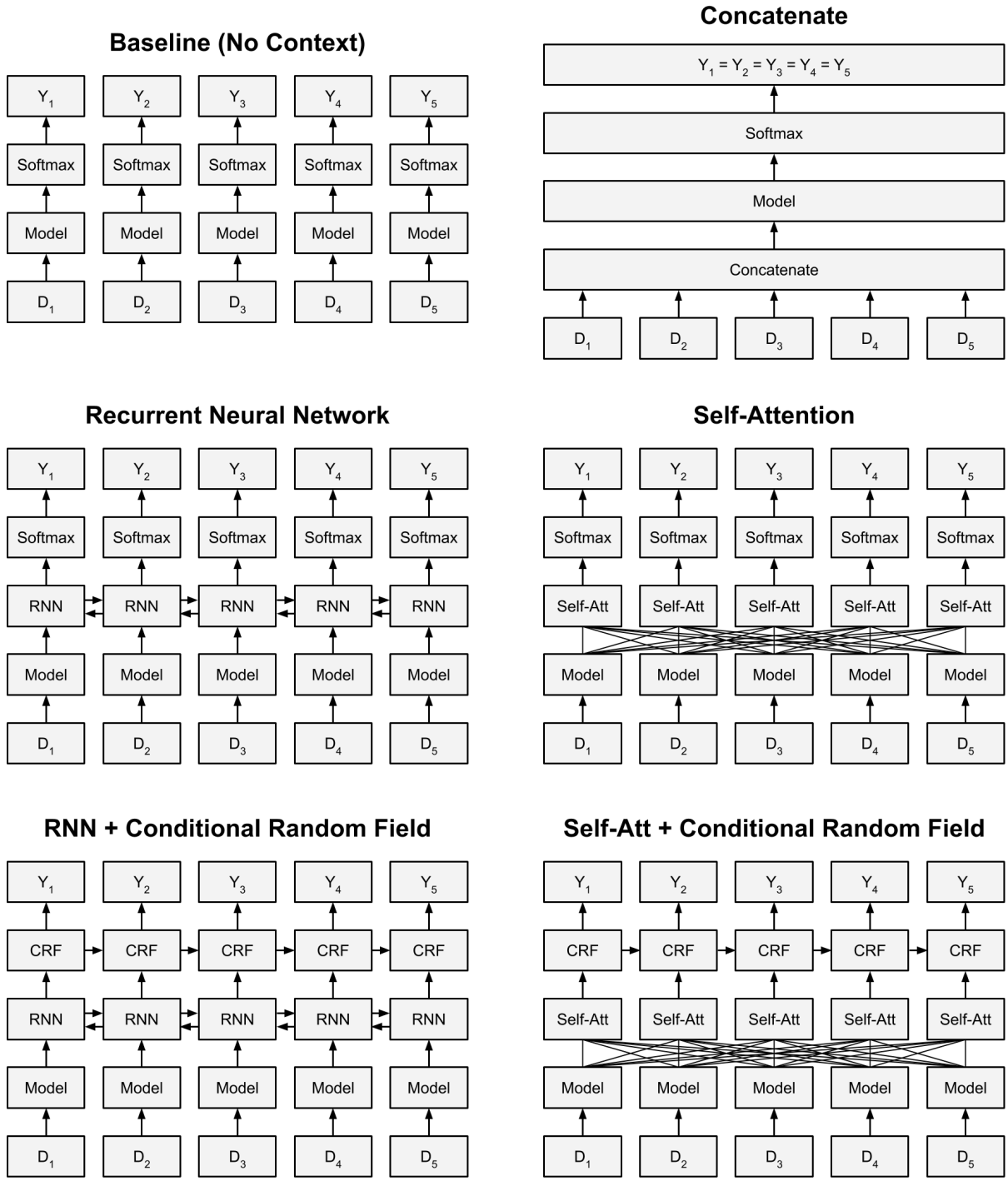


Figure 4.1: The baseline case for classifying EHRs and the five methods for incorporating case-level context from other reports. In the figures above, "Model" represents an arbitrary deep learning model designed for text classification, the output of which is an embedding representation of the input document.

This strategy is only valid under the condition that all reports within a given sequence share the same label; that is, $y_i = y_j$ for all i and j in the sequence. For example, in our application, all cancer pathology reports associated with the same unique tumor ID are tagged with the same aggregate-level labels. This strategy fails under the condition where each report in the sequence has a different label because the model would be forced to predict different labels from the same input.

Another notable limitation of concatenation is that it significantly increases the length of the input text that is fed into the model. Depending on the type of model used, this can cause severe problems. For example, RNN-based models are extremely slow and difficult to train when input sequences become too long [159, 125]; likewise, the memory required by self-attention-based models scales quadratically based off input length [130]. For long sequences where n is large, many models may become prohibitively expensive in terms of time and/or space complexity. In our experiments, we found that even the memory-efficient text CNN [68] has memory issues when the input sequence length exceeds 20K tokens, which was easily reached when concatenating sequences of 20+ pathology reports.

Recurrent Neural Networks

RNNs are a type of neural network architecture designed to process sequential information [82]. RNNs take in a series of inputs and produce a series of outputs. At any given timestep in the series, the output of the RNN depends not only on the input at the current timestep, but also on the inputs from all previous timesteps. This allows RNNs to recognize meaningful patterns over a sequence of entries, such as a series of EHRs over time.

The two most popular types of RNNs are long short-term memory (LSTMs) [50] and gated recurrent units (GRUs) [21]. Whereas more basic RNNs treat every entry in a sequence with equal importance, LSTMs and GRUs utilize gating operations to recognize when to save important information and when to skip less relevant entries; this allows LSTMs and GRUs to recognize more complex patterns over much longer sequences. In this work, we use GRUs because they have previously performed slightly better than LSTMs on EHRs and biomedical

text [58, 120]. The operations for a GRU are shown below:

$$
\begin{aligned}
z_t &= \sigma(W_z[h_{t-1}, x_t] + b_z) \\
r_t &= \sigma(W_r[h_{t-1}, x_t] + b_r) \\
c_t &= tanh(W_c[r \circ h_{t-1}, x_t] + b_c) \\
h_t &= (1 - z_t) \circ h_{t-1} + z \circ c_t
\end{aligned}
\tag{4.2}
$$

In the equations above, c_t is the processed value of the current input, which is a combination of the current input x_t and previous output h_{t-1}. r_t is a "reset gate" that controls the influence of the previous output h_{t-1} when calculating c_t. Finally, z_t is an "update gate" that determines how to combine c_t with the previous output h_{t-1} to generate the final output at the current timestep. Each operation relies on a function based on a learned weight W and bias b and the concatenation of the output from the previous timestep h_{t-1} and the input at the current timestep x_t.

To capture case-level context from EHRs, we utilize a GRU in conjunction with an existing deep learning text classification model designed to classify single reports, such as a text CNN [68]. Generally speaking, deep learning models designed for text classification will first encode a document into a final "document embedding", which is then passed onto a softmax layer for classification. The document embedding is usually generated by the penultimate layer of the deep learning model, and it represents the most important information used to classify a given document. Given a sequence of EHRs $d_0, ...d_i, ..., d_n$, we first use an existing deep learning model to generate document embeddings $e_0, ...e_i, ..., e_n$ for each report. We then feed these into a GRU (with optional bidirectionality) as follows:

$$
\begin{aligned}
o_i &= \text{BiGRU}(e_0, ..., e_i, ..., e_n) \\
y_i &= \text{Softmax}(W_s o_i + b_s)
\end{aligned}
\tag{4.3}
$$

where o_i is the ith output generated by the GRU. o_i is then fed into a softmax classifier or linear-chain CRF to generate the final label y_i. When making a decision for any given EHR, the GRU can take advantage of contextual information from other EHRs that came before (and in the case of bidirectionality, after) that report.

Self-Attention

Self-attention is a relatively new alternative to RNNs made popular by the Transformer architecture [141]. Like RNNs, self-attention takes in a series of inputs and generates a series of outputs; however, self-attention has been shown to both achieve higher accuracy and run faster than RNNs on a wide range of NLP tasks [27, 157, 138]. In our work, we use an implementation similar to that from the original Transformer paper, which is described below:

$$\begin{aligned} Q &= \text{ELU}(\text{Conv1D}(X + P, W_q) + b_q) \\ K &= \text{ELU}(\text{Conv1D}(X + P, W_k) + b_k) \\ V &= \text{ELU}(\text{Conv1D}(X + P, W_v) + b_v) \\ \text{Self-Attention}(Q, K, V) &= \text{softmax}(\frac{QK^T}{\sqrt{d}})V \end{aligned} \tag{4.4}$$

In the equations above, $X \in \mathbb{R}^{n \times d}$ is a matrix of the entries in the input sequence, where n is the length of the sequence and d is the dimension size of each entry. $P \in \mathbb{R}^{n \times d}$ are positional embeddings [43, 31] that represent the absolute position of each entry in the sequence – this simply allows the self-attention module to capture information about the order of the entries in the sequence. In our application, P is randomly initialized and learned through training. $X + P$ is fed into three parallel 1D-convolution operations (with a window size of one entry and exponential linear unit activation [23]) to extract three different feature representations of the input sequence – Q, K, and V. W_q, W_k, W_v, b_q, b_k, and b_v are the weights and biases associated with each 1D convolution. The dot product of Q and K forms a $n \times n$ similarity matrix which captures the relationships between each entry in the sequence. The final output is a new sequence $O \in \mathbb{R}^{n \times d}$ in which each entry has captured information from all entries in the original sequence related to that entry.

For our implementation, we also utilize the multihead variant of self-attention, which splits the self-attention operation into h parallel sub-attention operations. The inputs into self-attention are split across the d dimension such that $\{Q_i, K_i, V_i, \} \in \mathbb{R}^{n \times d/h}$; this enables

each sub-attention to focus on a different portion of the feature space and has been shown to give a slight boost to performance [141]:

$$\begin{aligned} \text{Multihead Self-Att}(Q, K, V) &= [h_1, ..., h_h] \\ \text{where } h_i &= \text{Self-Attention}(Q_i, K_i, V_i) \end{aligned} \tag{4.5}$$

Like in the case of RNNs, to capture case-level context from EHRs, we use self-attention in conjunction with an existing deep learning architecture for text classification. Given a sequence of EHRs $d_0, ...d_i, ..., d_n$, we first use an existing deep learning model to generate document embeddings $e_0, ...e_i, ..., e_n$ for each report. This creates the input matrix $E \in \mathbb{R}^{n \times d}$, which takes the place of X in Equations 4.4 and 4.5; the self-attention operations then allow for capture of contextual information from other EHRs in the sequence. The output from self-attention is fed into a final softmax layer or linear-chain CRF for classification.

Softmax vs. Linear-Chain Conditional Random Field

Our RNN and self-attention methods can utilize either a softmax or linear-chain CRF as the final layer for label generation. Incorporating a linear-chain CRF instead of a softmax after an RNN has previously been shown to improve performance on various general NLP sequence tagging tasks, such as in part-of-speech tagging and named entity recognition [54].

$$P(y_i) = \frac{\exp(W_s o_i + b_s)}{\sum \exp(W_s o_i + b_s)} \tag{4.6}$$

We use the standard implementation of softmax for our softmax layer, which is described in Equation 4.6. y_i is the label associated the ith report in a sequence, o_i is the RNN or self-attention output associated the ith report in a sequence, and W_s and b_s are the learned weight and bias parameters.

$$\begin{aligned} P(\bar{y}) &= \frac{\exp(W_c F(\bar{o}, \bar{y}))}{\sum \exp(W_c F(\bar{o}, \bar{y}))} \\ \text{where } F(\bar{o}, \bar{y}) &= F(y_{i-1}, y_i, \bar{o}, i) \end{aligned} \tag{4.7}$$

We use the standard implementation of a linear-chain CRF layer for our CRF layer, which is described in Equation 4.7. y_i is the label associated the ith report in a sequence, $\bar{y}$ is all labels associated with the sequence, o_i is the RNN or self-attention output associated the ith report in a sequence, $\bar{o}$ is all outputs associated with the sequence, and W_c are the learned weight parameters.

Compared to softmax, the main difference is that the linear-chain CRF utilizes a feature function $F(\bar{o}, \bar{y})$ rather than directly utilizing o_i. When predicting y_i, this feature function not only utilizes o_i to identify the correct label for y_i but also incorporates the transition probabilities between consecutive labels y_i and y_{i-1} in a sequence. For example, in our specific application of cancer pathology reports, all reports within the same sequence are tagged with the same labels; therefore, the CRF should learn that given the label y_{i-1} of the previous entry, the probability of y_i transitioning to a different label is extremely low.

Modular vs. End-to-End Training

Except for the concatenation method, all other methods to capture case-level context are modular in that they can be trained independently from an existing deep learning model for text classification in a two-step fashion. A user can choose an existing deep learning text classification model designed to classify single documents, train it on a corpus of EHR texts, and use the trained model to generate document embeddings for each EHR; then, the user can train our case-level context module (e.g., RNN or self-attention with or without CRF) independently on the resulting document embeddings. The benefit of modular training is that it eliminates the necessity of engineering the RNN/self-attention/CRF layers directly into an existing model architecture, which may potentially create overly cumbersome models that are computationally burdensome.

If desired, the RNN/self-attention/CRF layers can still be integrated directly into an existing text classification model such that training is end-to-end. We compare the performance of modular two-step training with end-to-end training using text CNNs and show that training the RNN, self-attention, and CRF layers in a modular fashion results in similar performance compared to end-to-end training.

4.4.3 Dataset

Our full dataset consists of 546,806 cancer pathology reports obtained from the Louisiana and Kentucky cancer registries of the National Cancer Institute (NCI) Surveillance, Epidemiology, and End Results (SEER) program following the appropriate IRB protocol. These cover cancer cases of all types from Louisiana residents spanning the years 2004-2018 and Kentucky residents spanning the years 2009-2018. Each pathology report is associated with a unique tumor ID that indicates the specific patient and tumor for the report – each tumor ID may be associated with one or more pathology reports. For example, a patient may have an initial test to check for cancer at a particular site, secondary tests of neighboring organs to see if the cancer has spread, and a followup test to see if the cancer has developed.

Each unique tumor ID is tagged with aggregate ground truth labels for six key data elements – site, subsite, laterality, histology, behavior, and grade. These ground truth labels were manually annotated by a human expert with access to all data relevant to each tumor ID; this includes radiology reports and other clinical notes not available in our dataset. The SEER cancer registries require that each individual cancer pathology report be labelled with the aggregate tags belonging to its associated tumor ID. Therefore, all pathology reports associated with the same tumor ID will have the same labels. Each pathology report is labeled with one of 70 possible sites, 314 possible subsites, 7 possible lateralities, 4 possible behaviors, 547 possible histologies, and 9 possible grades; a detailed breakdown of number of instances per label is available in Appendix C.1. A notable challenge in automated classification of cancer pathology reports, which is captured by our dataset, is identifying the correct aggregate-level labels for each report in a tumor ID sequence, even if some reports are addenda that may not contain the necessary information for all six data elements.

A large number of cancer pathology reports in our dataset are associated with tumor IDs that have only a single pathology report; in other words, these pathology reports do not have any case-level context because there is only a single report in the sequence. Because these reports do not require case-level context for analysis, they are filtered out of our dataset. After filtering, our dataset consists of 431,433 pathology reports and 135,436 unique tumor

IDs; on average, each tumor ID is associated with 3.2 pathology reports. A more detailed histogram of the number of reports per tumor ID is available in Appendix C.1.

To simulate a production setting in which a model trained on older, existing reports must make predictions on new incoming data, we split our dataset into train, validation, and test sets based off date. We first group pathology reports by tumor ID. If any tumor ID is associated with a report dated 2016 or later, all reports from that tumor ID are placed in our test set. On the remaining reports, we use 80:20 random splitting to create our train and validation sets, ensuring that reports from the same tumor ID are all placed in the train set or in the validation set without being split between the two. This yields a train set of 258,361 reports, a validation set of 64,906 reports, and a test set of 108,166 reports.

We apply standard text preprocessing techniques including lowercasing text, replacing hex and unicode, and replacing unique words appearing fewer than five times across the entire corpus with an "unknown_word" token. A more detailed description of our text cleaning process is available in Appendix C.2.

4.4.4 Baseline Models

To capture case-level context, our RNN-based and self-attention-based approaches work in conjunction with an existing deep learning text classification model, which is used to produce the document embeddings for individual pathology reports. For this study, we utilize two deep learning text classification models that have previously been shown to be highly effective for classifying cancer pathology reports – a CNN [116, 118] and a HiSAN [].

The CNN is an adaptation of the common word-level CNN used for general NLP tasks [68] – it examines combinations of three, four, and five consecutive words at a time and identifies the most salient word combinations for a given task. The HiSAN is a newer approach that utilizes a hierarchical structure based off self-attention to identify meaningful combinations of words in a document; compared to the CNN, the HiSAN can capture longer-distance word relationships that may be useful for a given task. To our knowledge, the HiSAN is the current state-of-the-art in cancer pathology report classification. Because the CNN and HiSAN were both developed on a similar dataset to ours, we use the exact same architecture

and hyperparameter settings as those described in the original publications; for additional details, we refer the reader to the original papers.

4.5 Experiments

4.5.1 Setup Details

Our experiments are designed to compare the performance of our five proposed methods to capture report level context under different scenarios. For each of these five methods, we test using both the CNN and the HiSAN as the baseline approaches. For the all methods other than concatenation, the CNN and HiSAN are first trained independently on the individual reports in our corpus (without case-level context), and then the resulting document embeddings are saved and used as input. We test performance on six classification tasks on our corpus – site, subsite, laterality, histology, behavior, and grade.

As described in our problem description, we test our methods under two conditions. In the first, for any given pathology report in a sequence of reports, each method can access other reports that came both before and after that report. In the second, each method can only access other reports that came before that report. For the concatenation method, this is achieved by concatenating only content from reports that came before the target report. For the RNN-based method (with and without CRF), we use a unidirectional RNN that can only access information from previous entries rather than a bidirectional RNN that can see both forward and backward. In the self-attention-based method (with and without CRF), we add a masking layer such that for any given entry in the sequence, self-attention will only find relationships between that entry and previous entries in the sequence.

We tune the hyperparameters of our RNN-based method and self-attention-based method using our validation set. For the RNN-based method, we use a GRU with hidden size 300, and for the self-attention based method, we use multihead self-attention with 300 dimensions and 6 heads. As we noted previously, concatenation can be prohibitively expensive for more complex models because the input documents can become very long. Therefore, we test the

concatenation method using the CNN baseline model only, as the HiSAN was unable to fit the concatenated documents into memory.

Except for concatenation, our approaches are designed to be modular in that they are trained separately from the baseline model used to generate document embeddings. As an additional experiment, we use the CNN baseline to compare the performance of the modular setup to an end-to-end setup in which we integrate the RNN/self-attention/CRF layers directly onto the end of the CNN and train the both parts together.

All methods are trained using a batch size of 64 and the Adam optimizer [69] with learning rate of 1E-4. For each method, we train on the train set and then measure accuracy on the validation set after each epoch. We stop training when the validation accuracy fails to improve for five consecutive epochs. We save the model parameters after the epoch with the highest validation accuracy and use those to evaluate on our test set.

4.5.2 Evaluation Metrics

For each of our six classification tasks, we evaluate performance using two metrics – accuracy and macro F-score. We calculate macro F-score as follows:

$$\begin{aligned} \text{Precision}_c &= \frac{\text{True Pos}_c}{\text{True Pos}_c + \text{False Pos}_c} \\ \text{Recall}_c &= \frac{\text{True Pos}_c}{\text{True Pos}_c + \text{False Neg}_c} \\ \text{F1 Score}_c &= \frac{2 \times \text{Precision}_c \times \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} \\ \text{Macro F1 Score} &= \frac{1}{n} \sum_{c=i}^{n} \text{F1 Score}_c \end{aligned} \tag{4.8}$$

where n is the total number of possible classes within a given classification task and c is a specific class.

In any given task, accuracy measures the overall performance of each classifier across all possible classes, and it does not disproportionally penalize the classifier for underperforming on any one particular class. We note that because in classification tasks such as ours in which each report is assigned to exactly one class, accuracy is the same as micro F-score.

On the other hand, macro F-score is heavily influenced by the performance on the minority classes. Therefore, macro F-score is an important metric because the distribution of label occurrences is highly skewed in many of our tasks – a more detailed breakdown of instances per label for each task is available in Appendix C.1. When extracting information from clinical reports, it is generally important to accurately identify occurrences of rare medical conditions even if they do not appear very often. For both accuracy and F-score, we establish 95% confidence intervals using a data bootstrapping procedure [36] that is described in greater detail in Appendix C.2.

4.5.3 Results

Our experimental results are displayed in Table 4.1 for the CNN baseline and in Table 4.2 for the HiSAN baseline. Across both the CNN and HiSAN baselines, all five methods of capturing case-level context achieve significantly better accuracy than the baseline of not utilizing any case-level context at all. In the unidirectional case where each classifier can only access context from previous reports, self-attention with linear-chain CRF achieves the overall best accuracy and macro F-scores. In the bidirectional case where each classifier can access both past and future reports, self-attention achieves the overall best accuracy while self-attention with linear-chain CRF achieves the best overall macro F-scores.

Across all methods, the unidirectional approach in which the network can only access earlier reports performs worse than the bidirectional approach in which the network can access both earlier and future reports. This makes intuitive sense because the unidirectional approaches have access to less information. In our particular application, the ground truth labels are identified based off all reports in the sequence; therefore, for any given report, future reports may be relevant for accurately predicting the ground truth label. Despite this, our results show that the unidirectional approaches still significantly outperform the baseline of no case-level context.

In Appendix C.3, we show the performance comparison of our modular methods with their end-to-end equivalents using the CNN baseline model. To attain the best performance in end-to-end training, we first pretrain the CNN portion of the model independently, then

Table 4.1: Accuracy and Macro F-Score (with 95% confidence intervals) of our different methods to capture case-level context on six different classification tasks using the CNN as the baseline. The top row is our baseline without any report level context, the middle group shows results of methods than can access both future and previous reports in a sequence, and the bottom group show results of methods that can only access previous reports in a sequence.

	Site	**Subsite**	**Laterality**	**Histology**	**Behavior**	**Grade**
CNN - Baseline	89.07	59.82	89.64	73.82	96.91	71.94
Accuracy	(88.91, 89.21)	(59.57, 60.05)	(89.49, 89.79)	(73.59, 74.03)	(96.82, 96.99)	(71.72, 72.15)
	56.22	24.33	46.91	22.79	76.06	73.72
Macro F-Score	(55.45, 56.83)	(23.92, 24.89)	(46.01, 47.83)	(22.42, 23.46)	(73.15, 78.51)	(72.24, 74.95)
CNN w/ Concat All	91.95	64.17	92.44	78.26	98.49	80.13
Accuracy	(91.74, 92.01)	(63.72, 64.19)	(92.28, 92.54)	(78.12, 78.54)	(98.40, 98.52)	(79.84, 80.22)
	58.62	22.57	**51.81**	21.67	74.58	75.21
Macro F-Score	(57.91, 59.14)	(22.08, 22.75)	**(51.19, 53.02)**	(21.23, 22.11)	(71.85, 77.44)	(74.74,79.72)
CNN w/ Bi-RNN	92.37	63.16	92.28	79.59	98.61	79.72
Accuracy	(91.87, 92.37)	(62.78, 63.68)	(92.01, 92.51)	(78.90, 79.63)	(98.48, 98.70)	(79.26, 80.00)
	62.14	27.42	49.89	32.29	73.83	79.03
Macro F-Score	(60.26, 62.66)	(26.16, 27.46)	(46.91, 50.33)	(30.65, 32.56)	(69.56, 78.16)	(78.66, 80.02)
CNN w/ Bi-RNN + CRF	92.26	63.03	92.29	79.27	98.64	80.66
Accuracy	(91.97, 92.45)	(62.65, 63.59)	(92.21, 92.69)	(78.62, 79.35)	(98.50, 98.71)	(80.53, 81.27)
	64.17	32.88	47.22	33.85	76.22	79.31
Macro F-Score	(63.89, 66.79)	(31.87, 33.49)	(44.75, 51.84)	(33.88, 35.78)	(74.28, 82.98)	(78.79, 80.20)
CNN w/ Self-Attention	**92.60**	**64.40**	**92.49**	**80.55**	98.73	**82.68**
Accuracy	**(92.32, 92.79)**	**(63.94, 64.84)**	**(92.22, 92.67)**	**(79.89, 80.66)**	(98.57, 98.78)	**(82.19, 82.87)**
	61.92	30.20	47.52	35.27	71.48	**82.55**
Macro F-Score	(60.75, 62.80)	(29.73, 31.20)	(46.36, 50.64)	(34.02, 35.73)	(70.06, 79.63)	**(81.59, 82.70)**
CNN w/ Self-Att + CRF	92.30	62.53	92.15	78.81	**98.79**	82.08
Accuracy	(92.12, 92.60)	(62.15, 63.06)	(91.95, 92.45)	(78.33, 79.07)	**(98.72, 98.92)**	(82.18, 82.87)
	65.41	**34.46**	49.29	**37.62**	**77.28**	81.27
Macro F-Score	**(64.67, 67.70)**	**(33.00, 34.62)**	(46.16, 53.53)	**(36.09, 37.81)**	**(74.97, 84.09)**	(80.79, 82.13)
CNN w/ Concat Previous	90.42	**62.20**	91.47	76.20	97.78	75.52
Accuracy	(90.34, 90.62)	**(61.94, 62.39)**	(91.29, 91.57)	(75.85, 76.26)	(97.73, 97.88)	(75.42, 75.84)
	56.53	22.25	47.43	20.41	67.44	77.62
Macro F-Score	(55.86, 57.11)	(21.90, 22.68)	(46.22, 48.10)	(20.15, 21.02)	(66.81, 70.65)	(73.61, 78.28)
CNN w/ RNN	90.60	61.88	91.43	76.01	97.96	76.49
Accuracy	(90.39, 90.91)	(60.99, 61.92)	(91.20, 91.73)	(75.55, 76.32)	(97.81, 98.07)	(76.15, 76.93)
	56.78	26.11	45.73	28.79	71.15	76.80
Macro F-Score	(55.68, 58.01)	(24.84, 26.10)	(44.30, 52.28)	(28.03, 29.77)	(69.74, 78.59)	(75.77, 77.22)
CNN w/ RNN + CRF	90.82	61.50	91.37	76.53	98.32	77.23
Accuracy	(90.56, 91.09)	(60.73, 61.63)	(91.25, 91.78)	(76.07, 76.85)	(98.18, 98.41)	(76.98, 77,72)
	60.19	30.24	**47.65**	32.57	73.05	76.11
Macro F-Score	(59.01, 61.86)	(29.71, 31.37)	**(45.04, 48.61)**	(31.21, 33.01)	(69.10, 78.35)	(75.92, 77.29)
CNN w/ Masked Self-Att	90.63	61.72	91.35	76.66	98.19	76.88
Accuracy	(90.36, 90.88)	(60.89, 61.82)	(90.90, 91.45)	(75.92, 76.71)	(97.91, 98.17)	(76.33, 77.13)
	59.48	29.42	47.44	30.67	71.33	76.69
Macro F-Score	(57.80, 60.40)	(27.78, 30.30)	(45.02, 49.31)	(29.53, 31.32)	(68.46, 77.09)	(75.68, 77.12)
CNN w/ M. Self-Att + CRF	**91.06**	62.00	**91.84**	**77.08**	**98.40**	**80.54**
Accuracy	**(90.88, 91.41)**	(61.55, 62.42)	**(91.38, 91.89)**	**(76.50, 77.27)**	**(98.32, 98.54)**	**(80.14, 80.88)**
	61.09	**30.98**	46.11	**33.95**	**78.66**	**79.92**
Macro F-Score	**(60.20, 63.52)**	**(30.71, 32.37)**	(44.62, 53.20)	**(32.86, 34.69)**	**(71.72, 80.88)**	**(79.09, 80.43)**

Table 4.2: Accuracy and Macro F-Score (with 95% confidence intervals) of our different methods to capture case-level context on six different classification tasks using the HiSAN as the baseline. The top row is our baseline without any report level context, the middle group shows results of methods than can access both future and previous reports in a sequence, and the bottom group show results of methods that can only access previous reports in a sequence.

	Site	Subsite	Laterality	Histology	Behavior	Grade
HiSAN - Baseline	90.06	61.94	89.97	75.00	96.88	73.10
Accuracy	(89.90, 90.20)	(61.71, 62.17)	(89.81, 90.12)	(74.78, 75.21)	(96.80, 96.96)	(72.87, 73.30)
	62.98	30.31	51.46	33.20	79.73	74.45
Macro F-Score	(62.07, 63.69)	(29.95, 31.10)	(50.64, 52.37)	(32.36, 33.88)	77.23, 81.89)	(72.80, 75.79)
HiSAN w/ Bi-RNN	92.71	67.07	93.11	80.50	98.86	84.37
Accuracy	(92.49, 92.96)	(66.83, 67.69)	(92.78, 93.26)	(80.01, 80.75)	(98.85, 99.04)	(84.50, 85.17)
	67.63	37.26	52.72	38.26	82.81	83.69
Macro F-Score	(65.69, 68.57)	(35.88, 37.69)	(51.24, 56.81)	(37.74, 39.77)	(77.36, 86.03)	(83.29, 84.82)
HiSAN w/ Bi-RNN + CRF	92.44	66.66	92.59	79.82	98.75	84.35
Accuracy	(92.25, 92.75)	(66.10, 66.98)	(92.34, 92.80)	(79.61, 80.34)	(98.61, 98.82)	(83.79, 84.46)
	67.92	39.54	53.17	41.62	83.42	83.80
Macro F-Score	(66.61, 69.39)	(37.81, 39.81)	(51.40, 56.83)	(39.75, 41.74)	(80.42, 86.70)	(83.00, 84.20)
HiSAN w/ Self-Attention	**93.03**	**68.03**	**93.48**	**81.03**	**98.98**	**85.72**
Accuracy	**(92.99, 93.47)**	**(67.72, 68.61)**	**(93.15, 93.62)**	**(80.64, 81.37)**	**(98.88, 99.06)**	**(85.44, 86.07)**
	68.04	39.01	**55.56**	38.70	85.98	**86.12**
Macro F-Score	(65.91, 68.25)	(37.44, 39.23)	**(51.97, 61.50)**	(38.06, 39.98)	(82.13, 89.89)	**(85.76, 86.57)**
HiSAN w/ Self-Att + CRF	92.52	66.83	92.80	80.36	98.96	84.97
Accuracy	(92.34, 92.83)	(66.54, 67.44)	(92.59, 93.05)	(80.01, 80.74)	(98.79, 98.99)	(84.44, 85.09)
	68.17	**40.70**	54.74	**43.12**	**87.67**	85.35
Macro F-Score	**(66.77, 69.66)**	**(39.59, 41.47)**	(52.77, 57.99)	**(42.58, 44.56)**	**(81.70, 89.35)**	(84.56, 85.51)
HiSAN w/ RNN	91.37	64.13	91.81	77.08	98.24	79.15
Accuracy	(91.18, 91.70)	(64.06, 64.96)	(91.71, 92.21)	(76.56, 77.30)	(98.14, 98.38)	(78.77, 79.49)
	63.59	34.50	46.81	33.42	79.54	79.22
Macro F-Score	(62.40, 65.41)	(32.61, 34.83)	(47.53, 51.81)	(33.19, 35.18)	(74.15, 82.77)	(78.60, 79.96)
HiSAN w/ RNN + CRF	91.92	65.56	92.38	77.76	98.61	81.80
Accuracy	(91.53, 92.03)	(65.14, 65.99)	(92.29, 92.78)	(77.43, 78.18)	(98.43, 98.66)	(81.82, 82.55)
	65.62	36.99	50.38	38.76	85.25	81.58
Macro F-Score	(64.84, 67.95)	(36.45, 38.43)	(49.79, 59.41)	(38.43, 40.45)	(77.71, 86.18)	(81.10, 82.32)
HiSAN w/ Masked Self-Att	91.50	64.82	91.94	77.54	98.20	79.38
Accuracy	(91.26, 91.77)	(64.56, 65.42)	(91.87, 92.37)	(77.13, 77.86)	(98.15, 98.39)	(79.12, 79.86)
	63.81	35.32	50.34	36.00	81.77	80.29
Macro F-Score	(62.66, 65.73)	(34.07, 35.88)	(49.09, 55.02)	(34.91, 36.93)	(76.55, 84.38)	(79.27, 80.55)
HiSAN w/ M. Self-Att + CRF	**92.11**	**65.57**	**92.66**	**79.22**	**98.85**	**83.64**
Accuracy	**(91.75, 92.24)**	**(65.47, 66.32)**	**(92.45, 92.92)**	**(78.74, 79.49)**	**(98.65, 98.85)**	**(83.10, 83.77)**
	65.69	**37.85**	**52.22**	**39.17**	**86.10**	**83.00**
Macro F-Score	**(64.82, 67.96)**	**(37.21, 39.17)**	**(50.93, 54.85)**	**(37.16, 39.28)**	**(81.15, 88.75)**	**(81.53, 82.86)**

train the entire model (both the CNN and RNN/self-attention/CRF portions) using end-to-end training. Therefore, the main difference between the modular training method and the end-to-end training method is that in the end-to-end method, the CNN weights can be further fine-tuned during the contextual training portion.

We see that compared to modular two-step training, end-to-end training is neither consistently better nor worse in terms of accuracy and macro F-score; performance varies by task. Across the different tasks and approaches, modular training usually achieves within 1% relative accuracy compared to end-to-end training. We believe that these results support the view that users can utilize our modular approaches for capturing case-level context and attain similar or better performance compared to a more complicated end-to-end approach with an equivalent architecture.

4.6 Discussion

As discussed in our methods section, deep learning approaches for text classification generally encode an input document into a document embedding representation, which is then used for classification purposes. Our methods to capture case-level context transform these document embeddings such that they account for information from other relevant reports in the sequence. We can visualize the document embeddings before and after our modular add-ons to better understand the transformations that are taking place.

In Figure 4.2, we show the document embeddings of our pathology reports on the site task generated by the HiSAN without case-level context (left) and the HiSAN with the self-attention method for capturing case-level context (right). The top pair of subfigures visualizes all document embeddings from our test set, colored by the ground truth organ system. We notice that clusters generated by the HiSAN with case-level context are slightly cleaner than the baseline HiSAN – there is less overlap between clusters and there are fewer subclusters within each organ system. This suggests that adding case-level context improves the HiSAN's ability to distinguish between pathology reports belonging to different organ systems.

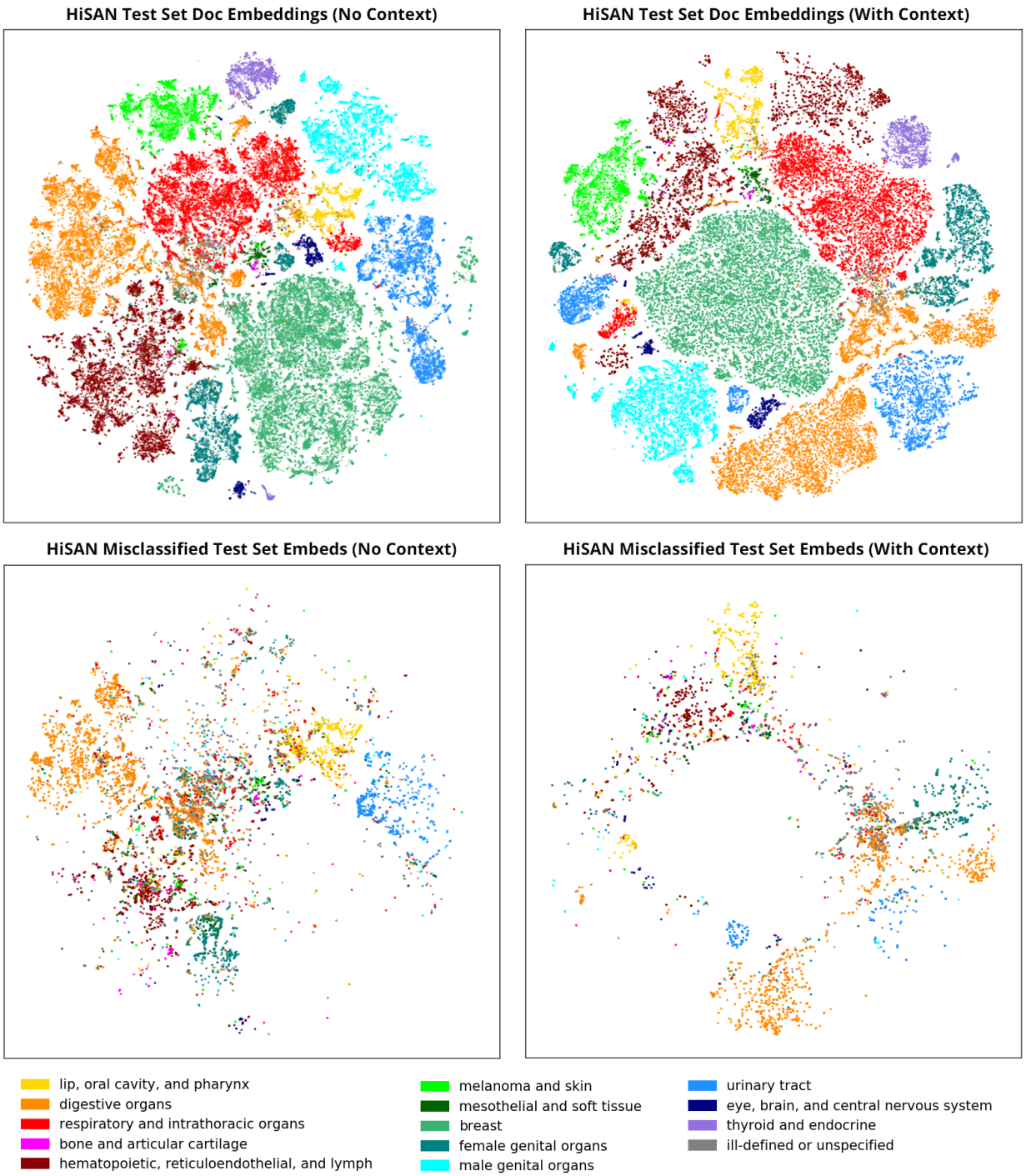


Figure 4.2: The top two subfigures show the cancer site document embeddings generated by the HiSAN for each pathology report in our test set with and without the self-attention module for capturing case-level context. The bottom two figures only show the document embeddings of misclassified reports in our test set. All document embeddings are colored by the ground truth organ system and visualized using t-SNE.

The bottom pair of subfigures show only the document embeddings of misclassified reports in the test set, colored by the ground truth organ system. This visualization allows us to better understand the types of errors that each approach makes. Based off the figure, we observe two general types of errors: (1) within-cluster misclassifications, in which the misclassified report is still clustered in the correct organ system, and (2) out-of-cluster misclassifications, in which the misclassified report is placed in an incorrect organ system. We see that adding document-level context reduces out-of-cluster errors compared to the baseline.

To gain a more in-depth understanding of the nature of the errors in our experiments, we randomly sampled 200 pathology reports that were misclassified by the baseline HiSAN (no case-level context) and manually examined the text of the pathology report. We then added the self-attention modular add-on and reclassified the same 200 reports to see which types of errors are resolved by incorporating case-level context.

Based off our manual examination, we identified two general categories of errors, which respectively correspond with the out-of-cluster and in-cluster misclassifications in Figure 4.2. In the first category of errors, the report either (1) does not appear to contain any information associated with the ground truth site or (2) mentions two or more (usually metastatic) sites; this is most likely because the report is an addendum or biopsy of a secondary or metastatic site. The baseline HiSAN therefore mispredicts the (non-ground truth) site that is mentioned in the report. Out of 200 randomly sampled reports, 80 reports fell into this category.

Adding case-level context can effectively deal with this type of error because the ground-truth label is almost always contained in another report in the sequence. Of the 80 reports misclassified by the baseline HiSAN in this first category, adding case-level context rectified 61 of the reports (76%).

In the second category of errors, the predicted site is a neighboring organ of the ground truth site or is within the same organ system as the ground truth site. Our manual analysis revealed that there is often overlap in the language used to describe organs within certain organ systems – for example, the ground truth site may be the rectosigmoid junction but the report may also mention the colon, or the ground truth site may be the cervix but the report may also mention the uterus. For these reports, we attempted to manually classify

the site ourselves without knowing the ground truth site or the HiSAN's predicted site, and more often than not we made the same prediction as the HiSAN; this indicates that language used in the reports is confusing not just for the HiSAN but also for an inexperienced human annotator. Four commonly confused groups of sites were (1) between C42 hematopoietic and reticuloendothelial systems, C44 skin, and C77 lymph nodes, (2) between C51 vulva, C52 vagina, C53 cervix, and C54 uterus, (3) between C64 kidney, C65 renal pelvis, C66 ureter, and C67 bladder, and (4) between C18 colon, C19 rectosigmoid junction, C20 rectum, and C21 anus.

This second category of errors also includes reports associated with ill-defined sites (C76), unknown sites (C80), or a general catch-all site for a particular organ system (e.g., C57 unspecified female genital organs). In these reports, the ground truth site is one of these ill-defined sites despite the report mentioning specific organs or cancer sites. Out of 200 misclassified reports examined, 120 reports fell into this second category.

Adding case-level context is less effective for dealing with this second category of errors because these confounding effects typically exist across all reports in the sequence; however, incorporating contextual clues from other reports may help narrow down the correct site. Of the 120 reports misclassified by the baseline HiSAN in this second category, adding case-level context rectified 35 of the reports (29%).

By visualizing the document embeddings from only the reports associated with a single tumor ID, we can show how adding case-level context affects the information captured in individual document embeddings. In Figure 4.3, we visualize the trajectories of the document embeddings belonging to four unique tumor IDs, colored by the *predicted* organ system. We see that the document embeddings generated by the HiSAN without case-level context are spread out over the embedding space – this is generally because each pathology report in a sequence may contain slightly different information, and as mentioned previously, multiple sites may be tested to check the spread of cancer to additional sites. Furthermore, there may be multiple different primary sites identified within the same tumor ID trajectory, likely because certain reports may contain information about secondary or metastatic sites. This is problematic because we wish to assign the same tumor-level labels to all reports belonging to the same tumor ID.

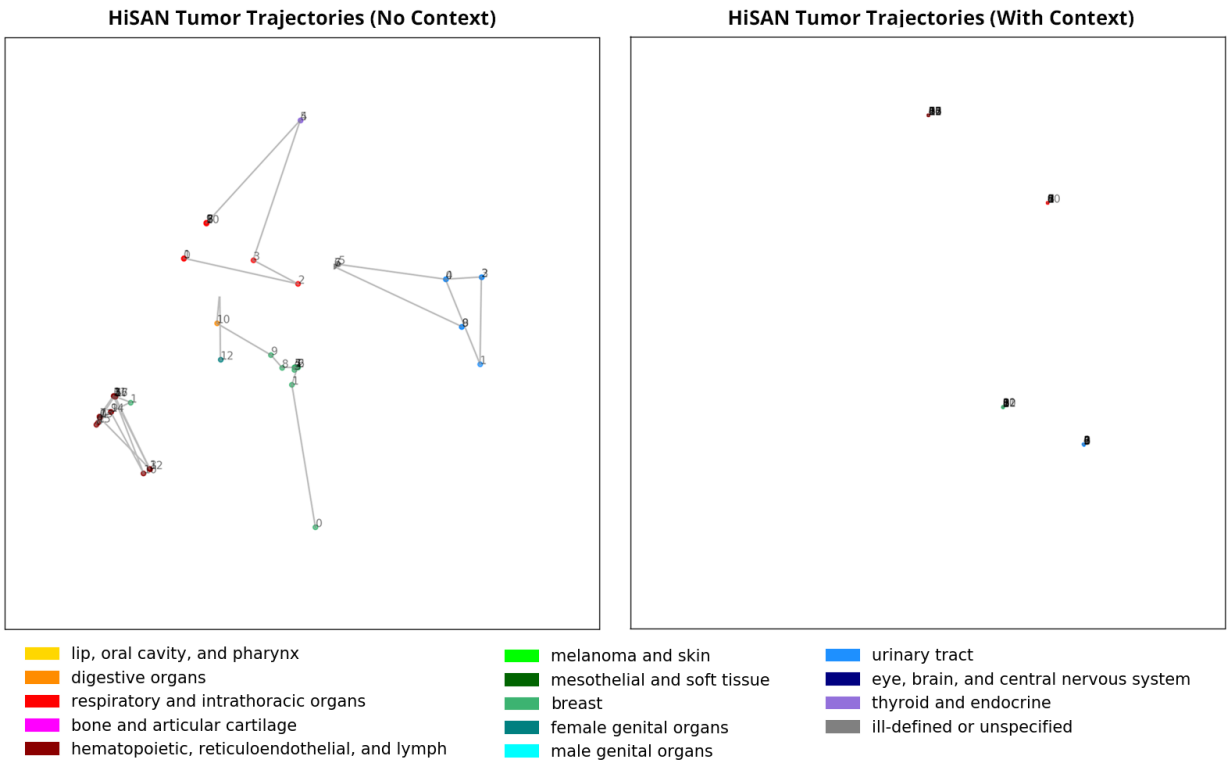


Figure 4.3: The cancer site document embeddings generated by the HiSAN for the pathology reports associated with four unique tumor IDs, with and without the self-attention module for capturing case-level context. These figures share the same axes as Figure 4.2 and thus can be directly compared. Within each of the four trajectories, document embeddings are numbered from earliest to latest and are colored by the predicted organ system. We notice that without case-level context, reports belonging to the same tumor ID are classified under different organ systems. Adding case-level context addresses this problem and all document embeddings from the same tumor ID are placed in the same location in the embedding space.

Once case-level context is incorporated, all document embeddings from the same tumor ID are placed in the exact same location – this is appropriate for our application because all pathology reports associated with the same tumor ID should have the exact same label. Furthermore, in the examples shown, all reports in the same trajectory are assigned the same label and thus misclassifications caused by secondary or metastatic sites are eliminated. We note that this type of trajectory analysis may be useful for identifying addendum-type and metastatic-type reports, which tend to be the pathology reports whose document embedding position shifts significantly and/or label changes once case-level context is included.

4.7 Conclusion

In this paper, we showed how adding a modular component for capturing case-level context on top of an existing deep learning text classification model designed for individual documents can improve classification accuracy of aggregate-level labels for cancer pathology reports. We compared the performance of five methods for capturing case-level context – concatenation, RNNs, RNNs with linear-chain CRF, self-attention, and self-attention with linear-chain CRF – and showed that all five achieved better accuracy than the baseline of no case-level context across six classification tasks. In the unidirectional case where each classifier can only access context from previous reports, self-attention with linear-chain CRF achieves the overall best accuracy and macro F-scores. In the bidirectional case where each classifier can access both past and future reports, self-attention achieves the overall best accuracy while self-attention with linear-chain CRF achieves the best overall macro F-scores.

Other than concatenation, our approaches are designed as modular add-ons that are easy to train on top of an existing deep learning text classification model built for individual documents. We show that our modular design, which uses a two-step training approach, has very similar performance to an identical end-to-end architecture, which requires far more engineering and may be prohibitively expensive in terms of time and memory for complex baseline models.

In our experiments, we demonstrated the effectiveness of our approach in the application for cancer pathology reports, where a sequence of reports belonging to a unique tumor

ID were all tagged with the same aggregate-level labels. We expect that with minimal modifications, our approaches can be applied towards a wide range of other EHR- and clinical text-based tasks. In future work, we plan to extend our experiments to clinical applications where each clinical report in a sequence is tagged with a different label, such as using a patient's previous clinical notes to inform the extraction of diagnosis or treatment codes from a given clinical report.

Chapter 5

Conclusion

5.1 Summary of Results

In Chapter 1, we proposed the following research questions:

1. Can hierarchical neural architectures, which have been shown to be highly effective in classifying general text such as news snippets and Yelp reviews, translate well into the clinical NLP domain and perform better than the existing state-of-the-art methods for classifying key data elements in cancer pathology reports?

2. What architectural improvements can be made to these hierarchical networks to adapt them to the nuances specific to clinical text and maximize their accuracy and speed?

In Chapter 2, we addressed our first research questions by introducing the Hierarchical Attention Network (HAN), which is a hierarchical RNN-based architecture originally developed in 2016 for classifying general text segments such as Yelp reviews and news snippets. We applied the HAN to cancer pathology report classification in order to show that it could beat out the previous state-of-the-art, CNNs, as well as other traditional machine learning classifiers trained on TF-IDF features such as logistic regression and support vector machines.

We tested the effectiveness of the HAN using a dataset of approximately 1000 cancer pathology reports of only breast and lung cancers obtained from five cancer registries (CT,

HI, KY, NM, Seattle). We evaluated on two tasks – subsite identification (12 possible subsites) and histology grade classification (4 possible grades); due to class imbalance, we used both accuracy (identical to micro F-score) and macro F-score for our evaluation metrics. Our results showed that the HAN achieves the best performance in both accuracy and macro F-score on both tasks, beating out all other baselines by a sizeable margin. Furthermore, our ablation studies showed that the hierarchical structure gives a very significant boost in performance compared to an RNN with attention, which is equivalent to the HAN without the hierarchical structure.

While chapter 2 demonstrated the effectiveness of HANs in terms of accuracy and macro F-score, the HAN still has a significant weakness – because its architecture is based off RNNs, it is very slow during training and inference, taking over ten times longer than a CNN. This can become noticeably burdensome when dealing with large amounts of training data, when computation resources are limited, or when a model must be repeatedly retrained when new data comes in. For example, we found that the HAN takes over a week to train on a corpus of approximately 235K pathology reports using a single GPU.

In Chapter 3, we addressed our second research question by introducing the Hierarchical Self-Attention Network (HiSAN), a novel hierarchical architecture based off self-attention and designed to overcome the slow training time associated with the HAN. The HiSAN uses a similar hierarchical structure as the HAN, but replaces the RNN layers with self-attention, which can capture long-distance word relationships like RNNs but runs much faster than RNNs for longer sequences. Using our task of classifying cancer pathology reports, we compared the performance of the HiSAN against HANs, CNNs, and several traditional machine learning classifiers in terms of accuracy, macro F-score, and training time.

We tested the effectiveness of the HiSAN using a dataset of approximately 375K cancer pathology reports covering all types of cancers obtained from the LA cancer registry. We evaluated using five tasks – site (70 possible labels), laterality (7 possible labels), behavior (4 possible labels), histology (516 possible labels), and grade (9 possible labels). Our results showed that the HiSAN achieves the best performance in both accuracy and macro F-score on all five tasks, achieving slightly better performance than the HAN. Similar to in Chapter 2, an ablation study showed that the hierarchical structure of the HiSAN gives a noticeable

boost in performance compared a similar architecture without the hierarchical structure. We also showed that the HiSAN performs especially well on the minority classes relative to the other classifiers – when we re-evaluated all classifiers using a limited dataset that excluded the top 15 most common cancer sites, the difference in performance between the HiSAN and the other baselines was much more noticeable.

In terms of timing, the HiSAN trained over ten times faster than the HAN, finishing training before the HAN completes a single epoch. Compared to the much more architecturally simple CNN, the HiSAN took about two to three times longer to train. We expect that our HiSAN architecture will enable users to reap the performance benefits associated with a using hierarchical structure while still maintaining a reasonable training time compared to a simpler model such as the CNN.

In Chapter 4, we further addressed our second research question by introducing hierarchical methods to classify cancer pathology reports that can make use of case-level context (i.e., contextual information from other related pathology reports from the same patient or cancer case), an application that has been previously unexplored in the clinical NLP domain. Given a sequence of cancer pathology reports associated with the same cancer case, our methods first utilize an existing deep learning classification model designed to operate on individual documents (such as a CNN or HiSAN) to generate document embeddings for each report; we then utilize a modular add-on to transform each document embedding into a context-aware embedding that includes useful information from other reports in the sequence. We compared our methods against the baseline of just using a deep learning classifier designed for individual documents (i.e., not using any case-level context) and of concatenating all reports from the same cancer case before feeding the reports into a deep learning classifier for individual documents.

We tested our methods using a dataset of approximately 550K cancer pathology reports covering all types of cancers obtained from the LA and KY cancer registries. We evaluated using six tasks – site (70 possible labels), subsite (314 possible labels), laterality (7 possible labels), behavior (4 possible labels), histology (547 possible labels), and grade (9 possible labels). Our results showed that our methods for capturing case-level context significantly beat out the baseline of not using any case-level context on all six tasks; in addition,

our methods for capturing case-level achieved better performance on almost all tasks than the baseline of concatenating reports before using a deep learning classifier for individual documents. Between our different methods explored for case-level context, the self-attention-based approaches achieved better overall performance and the RNN-based approaches.

5.2 Impact and Implications

The start-of-the-art techniques used in clinical NLP lags significantly behind those used in the general NLP community. According to a recent survey paper from 2018 [144], while there has been a huge surge in publications that utilize EHRs, the rate of publications on new clinical NLP techniques has remained relatively flat. Furthermore, rule-based approaches are still used in approximately 60% of the 400 papers surveyed – this contrasts sharply with the general NLP community, where out of 177 papers recently surveyed from the top 4 NLP conferences (NLP, EMNLP, ACL, and NAACL), only 6 used solely rule-based approaches [19]. The survey's authors suggest that a possible explanation for this is that NLP experts often have limited exposure to EHR data (partly due to the regulatory limitations regarding EHR privacy), while clinicians and domain experts are often untrained in advanced NLP techniques. Due to the existence of vast amounts of EHR data in the form of clinical text and importance of EHR-based applications, there may be many potential gains in applying more modern and powerful NLP techniques to clinical NLP tasks.

When it comes to deep learning for clinical NLP, the most popular approaches are very basic CNNs and RNNs dating back prior to 2015 [147]. In this work, we take steps toward bridging the gap between the state-of-the-art in clinical NLP and general NLP by introducing the HAN and HiSAN, which are more complex hierarchical neural architectures that borrow ideas from the state-of-the-art in general NLP and outperform basic CNNs and RNNs on classifying cancer pathology reports. The HAN and HiSAN both utilize a hierarchical structure that first breaks down a long document into smaller chunks and processes each smaller chunk individually before processing the document as a whole. The results from our ablation studies show that utilizing a hierarchical structure achieves better results than a similar architecture without the hierarchy, which suggest that this type of approach is better

at locating important information in long documents. This is especially relevant for clinical text, which can often be many pages long and have a relatively low signal-to-noise ratio for a given task.

The HiSAN utilizes self-attention layers in place of RNN layers, which allow it to achieve superior performance without being prohibitively slow to train – its training time is over 10x faster than the HAN, which has a similar hierarchical architecture but utilizes RNN layers. This is important for the democratization of AI because it allows users without access to powerful computational resources to still train and apply the HiSAN towards clinical text applications [137]; furthermore, given the high energy requirements for modern computing and GPU resources, faster training time is important for environment considerations [137]. Due to their faster training speed, HiSANs are also more practical for applications that require training and hyperparameter optimization on very large datasets or applications that need to retrain models on a regular basis in order to keep them up-to-date on new data.

Moreover, the HAN and HiSAN methods described in Chapter 2 and Chapter 3 enable better transparency of the workings within the deep learning models – the attention mechanisms utilized in the HAN and HiSAN show which words within a cancer pathology report are utilized most when making a classification decision. This can be a highly useful tool for human annotators when manually reviewing a pathology report – because the HAN/HiSAN highlights the keywords in each report associated with the target classification task, the human can focus on those sections of the report to reduce the amount of time required to come to a clinical decision.

In addition, Chapter 4 extends the concept of hierarchical structure to the document level in order to capture case-level contextual information when analyzing multiple EHRs in a sequence. One notable property of EHRs is that they are often sequential – a single patient or case may generate multiple reports over time. Within the same sequence, EHRs are generally related to each other in some manner; for example, the diagnosis of a disease in one EHR may prompt additional tests for that disease in following EHRs, and later EHRs may document the treatment or progression of the disease. In some applications, for any given clinical report, it is helpful or necessary to extract contextual information using other reports in the sequence. While there are many existing studies on how deep learning can

be applied on unstructured text and natural language from EHRs [147], to our knowledge, no studies have been done on how deep learning can utilize case-level context to capture aggregate information from a sequence of unstructured clinical text.

Developing more effective automated approaches for information extraction and classification of cancer pathology reports and other clinical text is critical for many important healthcare applications. For patient care, large scale automated processing of EHRs and other clinical reports could enhance patient safety and improve staff efficiency [121, 92]; for medical research, it can help improve our understanding of the impact of new diagnostics, treatments, and patient factors in patient outcomes for cancer and other diseases [92]. Better automated approaches for parsing EHRs may also help clinical decision support and be an important building block toward effective precision medicine [135]. In the context of the DOE-NCI partnership, we expect this work to be an important piece in the development of an automated system for classifying cancer pathology reports that can be deployed for practical use in cancer registries across the US and greatly reduce the burden on human annotators.

One notable limitation of this work is that our architecture development and experiments focused on the specific application of cancer pathology reports. We note that other unstructured clinical texts such as discharge summaries or radiology reports often share similar properties and challenges with cancer pathology reports – long length, low signal-to-noise ratio, and inconsistent grammar. Therefore, because our architectures deal well with these challenges, we believe the approaches developed in this work may be promising to many important clinical NLP applications beyond cancer pathology reports. However, because this is beyond the scope of this dissertation, we leave these experiments to future work.

5.3 Transfer Learning

Transfer learning is the practice in which a deep learning model that has been trained for a task is then transferred to a second different but similar task – once transferred to the second task, the original model can be directly applied to the second task or used as the

starting point for further training on the second task (often referred to as "fine-tuning"). If the first task is sufficiently related to the second task, the model may have learned useful features that are beneficial for the second task, thus giving a performance boost compared to training on the second task alone. For example, pretraining a deep neural network on the Image Net image classification dataset before fine-tuning on a downstream task such as object detection or action classification can yield notable performance improvements [56].

In our application of classifying cancer pathology reports, we have identified two potential use cases for transfer learning. First, collections of cancer pathology reports are generally distributed across various cancer registries, and each registry generally collects reports specific to its geographic region. A cancer registry may wish to develop a robust classification model for its cancer pathology reports but may not have access to the pathology reports from other cancer registries. Transfer learning would enable the registry to download a pretrained model (such as the models developed in this work) and then fine-tune the model on their own data, which may yield better performance than training a model from scratch on their own data.

Secondly, a particular application may need to detect a specific type of cancer or a particular cancer property. Because labelled datasets for clinical data are often relatively small in size, it may be more effective to download a pretrained model (such as the models developed in this work) and then fine-tune on a labelled dataset than train a model from scratch. For example, if an application needs only to detect different subsites of lung cancer, it may be more effective to download a model pretrained to identify all cancer types and then fine-tune on a limited dataset of lung cancer than to train a model on lung cancer from scratch.

We use our HiSAN architecture from Chapter 3 to perform two sets of transfer learning experiments that reflect the two use cases described previously. Using our full corpus of cancer pathology reports obtained from the LA and KY registries, we first split our dataset by registry (dataset statistics are described in Table 5.1) and explore how transfer learning may improve performance when pretraining on one registry and then fine-tuning on another.

Table 5.2 shows the performance of transfer learning when testing on the LA registry and Table 5.3 when testing on the KY registry. We see that when testing on the LA dataset, if

Table 5.1: Dataset descriptions for our transfer learning experiments. We split our full LA + KY dataset in two ways: (1) by registry and (2) top 15 most prevalent cancer sites vs minority cancer sites (all other sites). For each dataset, we show the number of samples in our train, validation, and test sets as well as the number of classes per task.

Dataset	Train Size	Val Size	Test Size	Site	Subsite	Lat.	Hist.	Beh.	Grade
LA+KY	335650	84143	127013	70	314	7	547	4	9
LA	236426	59540	78860	70	306	7	516	4	9
KY	99224	24603	48153	69	299	7	477	4	8
LA+KY (Top 15 Sites)	279800	70369	108143	15	84	7	402	3	9
LA+KY (Minority Sites)	55850	13774	18870	55	230	7	369	4	8

Table 5.2: Accuracy and Macro F-Score (with 95% confidence intervals) of the HiSAN on the LA dataset using different training regiments.

	Site	Subsite	Laterality	Histology	Behavior	Grade
Train on LA	90.34	62.99	89.34	76.06	96.60	71.03
Accuracy	(91.13, 90.54)	(62.66, 63.31)	(89.12, 89.55)	(75.77, 76.36)	(96.48, 96.72)	(70.70, 71.35)
	62.70	30.24	48.73	32.16	86.05	73.49
Macro F-Score	(61.56, 64.08)	(29.73, 31.38)	(47.47, 49.93)	(31.62, 33.79)	(83.84, 88.05)	(71.72, 74.78)
Train on KY	88.50	60.62	88.18	72.78	95.83	67.51
Accuracy	(88.29, 88.72)	(60.25, 60.95)	(87.95, 88.41)	(72.47, 73.07)	(95.69, 95.97)	(67.19, 67.84)
	63.42	29.75	49.80	26.96	83.06	65.68
Macro F-Score	(62.18, 64.35)	(29.07, 30.87)	(48.67, 50.88)	(26.73, 28.54)	(80.67, 85.24)	(62.64, 68.23)
Train on KY, Fine-Tune on LA	**90.39**	**64.06**	**89.35**	**76.41**	**96.67**	**71.24**
Accuracy	(90.19, 90.59)	(63.71, 64.39)	(89.13, 89.57)	(76.11, 76.70)	(96.54, 96.79)	(70.94, 71.56)
	64.43	**31.09**	**49.68**	**33.43**	**86.42**	**73.67**
Macro F-Score	(63.31, 65.81)	(30.60, 32.34)	(48.51, 50.82)	(32.88, 35.08)	(84.04, 88.59)	(71.92, 75.11)

Table 5.3: Accuracy and Macro F-Score (with 95% confidence intervals) of the HiSAN on the KY dataset using different training and transfer learning regiments.

	Site	Subsite	Laterality	Histology	Behavior	Grade
Train on LA	**90.21**	62.48	90.18	73.45	97.24	71.80
Accuracy	(89.94, 90.47)	(62.07, 62.91)	(89.92, 90.45)	(73.06, 73.81)	(97.08, 97.38)	(71.39, 72.21)
	64.11	29.61	49.80	30.24	84.37	61.41
Macro F-Score	(62.27, 65.20)	(28.71, 30.93)	(48.22, 51.29)	(29.66, 32.07)	(81.72, 86.85)	(60.63, 62.15)
Train on KY	89.81	62.95	90.11	73.24	97.18	73.30
Accuracy	(89.53, 90.08)	(62.51, 63.37)	(89.83, 90.37)	(72.84, 73.63)	(97.04, 97.33)	(72.92, 73.69)
	63.10	31.10	50.17	28.75	86.44	62.96
Macro F-Score	(61.61, 64.54)	(30.51, 32.67)	(48.67, 51.67)	(28.22, 30.50)	(83.79, 88.75)	(62.27, 63.75)
Train on LA, Fine-Tune on KY	**90.21**	**63.47**	**90.19**	**74.55**	**97.26**	**73.94**
Accuracy	(89.93, 90.48)	(63.05, 63.90)	(89.93, 90.45)	(74.14, 74.95)	(97.12, 97.41)	(73.54, 74.33)
	64.44	**33.19**	**49.89**	**33.22**	**86.06**	**64.22**
Macro F-Score	(62.40, 65.72)	(32.24, 34.69)	(48.26, 51.38)	(32.58, 35.05)	(83.45, 88.33)	(63.54, 64.92)

the model pretrained on the KY registry is directly applied to the LA dataset without fine-tuning, it performs worse than a model trained from scratch on the LA registry. However, this is not the case when testing on the KY registry – when the model pretrainined on the LA registry is directly applied to the KY dataset without fine-tuning, it outperforms the model trained from scratch on the KY dataset on some tasks. We expect that this is simply because the LA dataset is much larger and therefore models trained on the LA dataset may be more robust on some tasks than those trained on the KY dataset. For both registries, we see that when the pretrained model is fine-tuned on the target registry, it slightly outperforms the model trained from scratch on the target registry on all tasks.

In our second set of experiments, we test the effectiveness of transfer learning when pretraining on a large dataset containing one set of cancer types and then fine-tuning on a much smaller dataset containing a separate set of cancer types. For this set of experiments, we split our full dataset into two subsets – one which contains only reports associated with the 15 most prevalent cancer sites and one which contains the remaining reports from the minority cancer sites (dataset statistics are described in Table 5.1). The top 15 most prevalent sites in order from most to least prevalent are C50 breast, C34 lung, C42 hematopoietic and reticuloendothelial systems, C18 colon, C61 prostate, C44 skin, C77 lymph nodes, C67 bladder, C54 corpus uteri, C20 rectum, C73 thyroid, C16 stomach, C64 kidney, C25 pancreas, and C56 ovary.

The results of our second experiment are shown in Table 5.4. We see that when the model trained on the top 15 cancer sites is applied directly on the minority site dataset without fine-tuning, it completely fails in the site and subsite tasks and underperforms in the other tasks. This makes sense because the model has never seen any of the sites or subsites in the minority class dataset; furthermore, the criteria for identifying the labels for the other tasks is often dependent on the cancer site. However, once fine-tuning is added, the fine-tuned model performs slightly better across most tasks than the model trained from scratch on the minority sites.

Our experiments show that transfer learning can be used to give a slight performance boost when working with cancer pathology reports. When the available dataset is limited in size or scope, downloading a pretrained model that has been trained on a much larger

Table 5.4: Accuracy and Macro F-Score (with 95% confidence intervals) of the HiSAN on only the minority cancer sites (cancer sites excluding the top 15 most common cancer sites) from the LA+KY dataset. We compare performance using different training and transfer learning regiments.

	Site	Subsite	Laterality	Histology	Behavior	Grade
Train on Minority Sites	76.95	61.97	84.29	71.10	96.28	70.05
Accuracy	(76.36, 77.57)	(61.29, 62.69)	(83.74, 84.81)	(70.42, 71.74)	(96.02, 96.54)	(69.39, 70.70)
	65.47	**30.80**	32.55	32.39	84.13	65.93
Macro F-Score	(63.74, 66.89)	(30.08, 32.23)	(32.20, 32.92)	(32.08, 34.88)	(82.30, 85.94)	(61.94, 68.85)
Train on Top 15 Sites	00.00	00.00	62.52	49.69	90.78	64.09
Accuracy	(00.00, 00.00)	(00.00, 00.00)	(61.82, 63.22)	(48.94, 50.37)	(90.36, 91.17)	(63.40, 64.76)
	00.00	00.00	24.54	8.86	38.64	53.52
Macro F-Score	(00.00, 00.00)	(00.00, 00.00)	(23.69, 25.41)	(8.82, 9.87)	(37.98, 39.26)	(51.46, 55.35)
Train on Top 15, Fine-Tune on Minority	**77.38**	**62.55**	**84.63**	**71.86**	**96.55**	**71.28**
Accuracy	(76.79, 77.97)	(61.82, 63.22)	(84.09, 85.14)	(71.20, 72.48)	(96.29, 96.80)	(70.61, 71.92)
	64.71	30.24	**33.30**	**33.39**	**85.66**	**73.04**
Macro F-Score	(62.94, 66.53)	(29.69, 31.82)	(32.52, 34.46)	(32.89, 35.82)	(83.89, 87.23)	(69.66, 75.57)

corpus and then fine-tuning on the available data may yield slightly better performance than training on the available data alone.

5.4 Applications to General Text

In designing the architecture for our HiSAN, we specifically focused on the application of cancer pathology reports, which have characteristics unique to clinical text such as long length and low signal-to-noise ratio. However, we found that while the HiSAN offered superior performance on cancer pathology reports, performance could be further improved on classification tasks on shorter general text segments (e.g., Yelp or Amazon reviews) by adding additional complexity to the network. This led to the development of a model similar to the HiSAN, the Hierarchical Convolutional Attention Network (HCAN), which shares the same hierarchical structure as the HiSAN and also shares many of the same architectural components.

The architecture of our HCAN network is shown in Figure 5.1. The key differences between the HCAN and the HiSAN are the use of a temporal convolution layer before self- and target-attention, positional embeddings, layer normalization, and two parallel self-attention layers. For more information about each of these changes, we refer the reader to

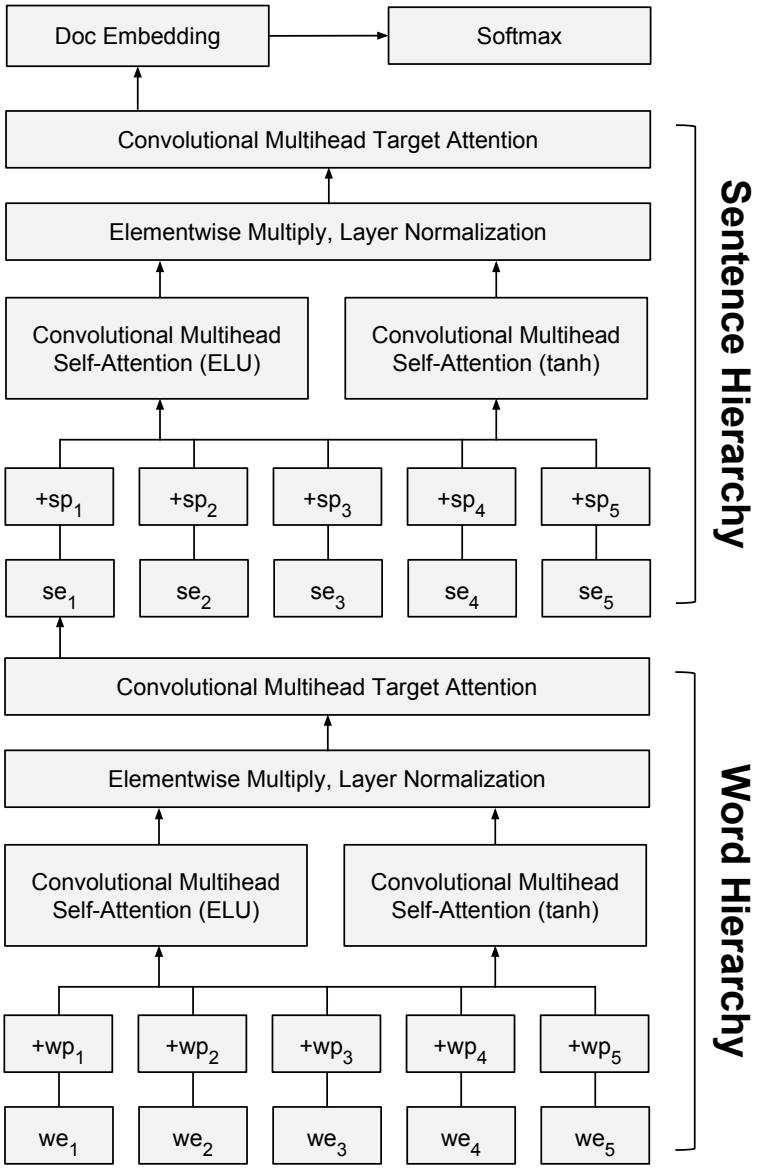


Figure 5.1: The architecture for our Hierarchical Convolutional Attention Network.

our published paper on the HCAN, "Hierarchical Convolutional Attention Networks for Text Classification" [41].

We tested the performance of our HCAN using four text datasets which are described in Table 5.5 and in greater detail in our full paper. We compare against traditional machine learning classifiers and a basic CNN architecture; we also performed various ablation tests to show how each additional component contributed to the final performance. Our results are summarized in Table 5.6.

Just as the HiSAN achieves superior performance in cancer pathology reports, we see that the HCAN achieves the best accuracy across all four general text classification tasks compared to our baselines. In terms of timing, the HCAN trains about 2-3 times as fast as the HAN depending on the task (Table 5.6, Figure 5.2), which is again similar to our results from the HiSAN. We note that the relative improvement in training time for the HCAN on general text is less than the relative improvement for the HiSAN on cancer pathology reports because the general text used in the HCAN experiments is typically no longer than a single paragraph. While RNN-based approaches like the HAN are generally slower to train than self-attention-based approaches like the HCAN and HiSAN, training time for RNNs scales nonlinearly based off input document length so they are less prohibitively slow on shorter texts.

The results from our HCAN experiments suggest that variations of our hierarchical approaches approaches may bring about improvements in NLP applications across domains other than cancer pathology reports and clinical text – these applications may be explored further in future work.

Table 5.5: Dataset Descriptions

Dataset	Classes	Documents	Vocabulary	Task Description
Yelp Reviews 2016	5	1,033,037	72,880	Sentiment Analysis
Amazon Reviews Sentiment	5	500,000	67,802	Sentiment Analysis
Amazon Reviews Category	10	500,000	67,802	Topic Classification
Pubmed	8	800,000	182,167	Topic Classification

Table 5.6: Test set accuracy, mean training time for a single document, and total training time on each task.

Classifier	Yelp 2016	Amazon Sentiment	Amazon Category	Pubmed
Naive Bayes	63.12 –, 1.8s	61.66 –, 0.8s	88.14 –, 1.3s	75.81 –, 4.2s
Logistic Regression	71.31 –, 306s	67.57 –, 101s	88.69 –, 173s	78.57 –, 463s
Word shallow-and-wide CNN	74.44 17ms, 9hr	70.75 15ms, 5hr	88.20 15ms, 5hr	78.15 35ms, 22hr
Hierarchical Attention Network	76.30 96ms, 67hr	72.56 97ms, 35hr	89.68 113ms, 37hr	79.89 167ms, 110hr
Conv Attention Network (One Self-Attention, Maxpool)	75.01 19ms, 13hr	71.24 19ms, 8hr	89.27 19ms, 8hr	79.21 38ms, 25hr
Conv Attention Network (One Self-Attention, Target Attention)	75.17 21ms, 14hr	71.45 21ms, 9hr	89.35 21ms, 9hr	79.70 39ms, 26hr
Conv Attention Network (Two Self-Attentions, Maxpool)	75.21 23ms, 15hr	71.45 22ms, 9hr	89.41 22ms, 9hr	79.86 41ms, 27hr
Conv Attention Network (Two Self-Attentions, Target Attention)	75.25 25ms, 17hr	71.78 24ms, 10hr	89.71 24ms, 10hr	79.95 43ms, 29hr
Hiearchical Conv Attention Network (One Self-Attention, Maxpool)	75.00 24ms, 16hr	71.09 23ms, 9hr	88.85 23ms, 9hr	79.31 42ms, 28hr
Hierarchical Conv Attention Network (One Self-Attention, Target Attention)	75.75 34ms, 23hr	72.33 29ms, 12hr	89.55 29ms, 12hr	79.91 47ms, 31hr
Hiearchical Conv Attention Network (Two Self-Attentions, Maxpool)	75.54 31ms, 21hr	72.43 31ms, 13hr	89.34 31ms, 13hr	80.09 50ms, 33hr
Hierarchical Conv Attention Network (Two Self-Attentions, Target Attention)	**76.51** 49ms, 32hr	**72.85** 38ms, 16hr	**89.89** 38ms, 16hr	**80.13** 53ms, 35hr

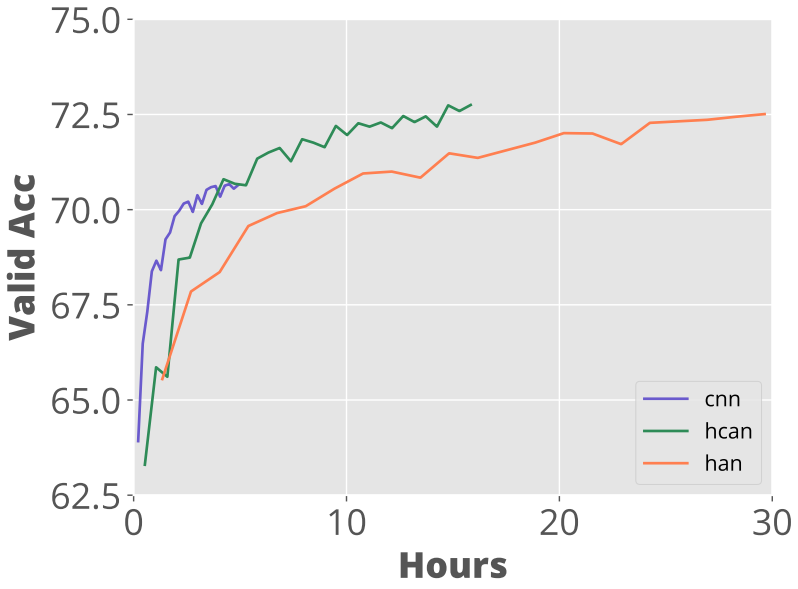


Figure 5.2: Validation accuracy vs time on Amazon sentiment analysis task.

5.5 A Note on ELMo and BERT

In 2017, the seminal paper Embeddings from Language Models (ELMo) [112] initiated a paradigm shift within the NLP community – this paper made popular the use of contextual word embeddings, which are word embeddings that adapt to the context in which words appear. ELMo utilizes layers of bidirectional RNNs to process an input sentence or document; the hidden states or outputs from the RNNs, which capture contextual linguistic information about each word, can then be extracted and used as the word embedding inputs for a downstream model. While this idea is not entirely new (similar ideas were explored previously in [90] and [111]), ELMo was the first paper to show that this approach could yield significant gains across a wide range of different NLP tasks, such as question answering, textual entailment, and sentiment analysis. Part of this success was due to the fact that ELMo was pretrained on the enormous 1B Word Benchmark dataset [17] before being applied to downstream tasks, enabling it to learn useful linguistic features beyond what can be learned from training on only the (potentially limited) dataset associated with a downstream task alone.

Bidirectional Encoder Representations from Transformers (BERT) [28] took the ideas from ELMo one step further – whereas ELMo only pretrains contextual word embeddings which are then fed into a downstream model, BERT pretrains an entire neural architecture which is then fine tuned on a specific downstream task. In other words, whereas ELMo requires a downstream model to process its contextual word embeddings, BERT includes the downstream model in its pretraining.

The BERT architecture is a deep neural architecture that utilizes 8-16 self-attention layers (similar to those used in the HiSAN or HCAN) with a hidden size of 768-1024 per layer for total of approximately 110M to 340M learnable parameters. The core idea of BERT is to pretrain the network weights using a very large amount of unlabelled text – the original paper pretrains on the BooksCorpus and English Wikipedia, which combined is approximately 3.5B words. Two unsupervised pretraining tasks are used to pretrain BERT: (1) given a sentence input, random words are dropped and must be reconstructed based off the words present and (2) given two sentences, the network must identify if they are

neighboring sentences from the same text. Like in ELMo, the large number of learnable parameters combined with pretraining on a large amount of unlabelled data allows BERT to learn useful linguistic features that may not be present if limited only to the dataset specific to a downstream NLP task.

When it was first released in 2018, BERT achieved new state-of-the-art results a wide range of NLP tasks including sentiment analysis, natural language inference, textual entailment, and paraphrasing. XLNet [151], released in 2019, took the core principles of BERT and introduced an improved training method which resulted in further improvements in performance across many NLP tasks. The current state-of-the-art in many NLP applications often incorporate some version of BERT or XLNet into their pipeline; for example, in the iconic Stanford Question Answering task[1], most of the top 10 solutions are based off BERT or XLNET.

Prior to BERT, many performance gains in various NLP tasks were achieved solely through architectural innovations and improvements; however, the BERT has shown that pretraining using a large corpus of unlabelled text and then transferring that knowledge to a downstream task through fine tuning can yield much larger improvements. In other words, using a large amount data for pretraining combined with a large, complex model is better than just a large, complex model.

To establish a performance benchmark of the hierarchical architectures introduced in this work, we compare the performance of our HCAN against BERT on the Yelp '14 dataset based off the results described in [2]. Our results are shown in Table 5.7. Surprisingly, our HCAN achieves competitive performance, lagging behind BERT (large) by only 0.3% in test accuracy. We note that while we attempted to match the preprocessing and data setup described in [2], there may still be minor differences between our implementations. However, our results suggest that at least when it comes to text classification (which is oftentimes less complex than other NLP tasks like machine translation or question answering and therefore may require less complex architectures), our hierarchical architectures offer strong performance without the increased training time and need for unlabelled pretraining data associated with an approach like BERT.

[1] https://rajpurkar.github.io/SQuAD-explorer/

Table 5.7: Validation and Test Accuracy of different classification models on the Yelp '14 sentiment analysis task. All results other than HCAN are obtained from [2].

Model	Yelp '14 Val. Acc.	Yelp '14 Test Acc.
LR	61.1	60.9
SVM	59.7	59.6
CNN	66.5	66.1
LSTM	69.0	68.7
HAN	–	70.5
BERT (base - 8 layers x 768)	72.1	72.0
BERT (large - 16 layers x 1024)	**72.6**	**72.5**
HCAN	71.9	72.2

While BERT and XLNet have been highly successful across a wide range of general NLP tasks, its use in the clinical domain is still limited, likely for the following reasons:

- BERT is an extremely large model and is thus very slow to train. Based off one estimate[2], it would take 34-58 days to train BERT (based off the setup described in the original paper) using a standard desktop with four RTX 2080 Ti GPUs. For clinical NLP practitioners with limited computational resources, this makes BERT impractical for many applications.

- Because BERT has 110M-340M learnable parameters, it requires a very large amount of data for pretraining on the order of 1B+ words. For general NLP tasks, this data is easily available in the form of web data, news articles, social media posts, or Wikipedia articles. For clinical NLP tasks, this data can be more difficult to come by, not only because access to data is restricted by regulatory policies, but also because data is often decentralized across different organizations and institutions.

- While pretrained BERT models are available for download, these are generally pretrained on general text which is linguistically different from clinical text. Studies have shown that the domain used for pretraining is an important factor in how well BERT performs on a downstream task [5, 74, 8].

[2] https://timdettmers.com/2018/10/17/tpus-vs-gpus-for-transformers-bert/

- The original BERT architecture is designed to process relatively short text segments such as sentences. While this is appropriate for some clinical NLP tasks, such as medical natural language inference, it may be less suitable for tasks that utilize long documents as input such as classifying cancer pathology reports. Some additional work may be required to effectively adapt BERT to train and predict on these tasks.

We believe that an architecture similar to BERT, if successfully implemented on cancer pathology reports and given a training corpus of at least 1-10 million pathology reports or similar clinical documents, may bring about additional improvements in accuracy relative to our HiSAN at the cost of significantly increased training and inference time. However, we leave the full implementation of such a study for future work because it is beyond the scope of this dissertation.

5.6 Additional Future Work

Based off the work done in this thesis, we have identified three further avenues of research that we believe may bring about additional improvements to the methods described in this work – (1) uncertainty quantification, (2) semi-supervised learning, and (3) multitask learning.

Although we show that the methods introduced in this work achieve higher accuracy and macro F-score across all tasks compared to previous baselines, there is still room for improvement on the difficult tasks, such as histology and grade, in order to bring it to a level in which it may be deployed in place of a human expert. For these more difficult tasks, uncertainty quantification methods may be utilized so that predictions with high confidence can be used with minimal human review, thus reducing the burden on human annotators.

The methods presented in this work are all supervised methods in that they operate on labelled data. However, when it comes to EHRs and clinical text, labelled data is often difficult to come by because it is labor-intensive to produce and may require expensive expert knowledge. Since unlabelled clinical text and EHR data is generally more widely available than labelled data, incorporating semi-supervised approaches that can effectively learn from unlabelled EHR and clinical text data may further boost the performance of the methods presented in this work.

Finally, in many clinical NLP applications, there may be multiple tasks per input document. For example, in this work, for each cancer pathology report, we must extract multiple data elements including site, laterality, histology, behavior, and grade. The work in this thesis focuses on single-task approaches, in which a separate model is trained on each task. However, this does not allow for information-sharing between tasks – features that are used in one task may also be helpful for another task. We expect that further exploration of multitask approaches may be beneficial in exploiting these relationships between tasks and provide additional gains in performance.

Bibliography

[1] Adams, K. T., Howe, J. L., Fong, A., Puthumana, J. S., Kellogg, K. M., Gaunt, M., and Ratwani, R. M. (2017). An analysis of patient safety incident reports associated with electronic health record interoperability. *Applied Clinical Informatics*, 8(2):593–602. 1

[2] Adhikari, A., Ram, A., Tang, R., and Lin, J. (2019). Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*. xi, 114, 115

[3] Alawad, M., Gao, S., Qiu, J. X., Yoon, H.-J., Christian, J. B., Penberthy, L., Mumphrey, B., Wu, X.-C., Coyle, L., and Tourassi, G. (2019). Automatic extraction of cancer registry reportable information from free-text pathology reports using multi-task convolutional neural networks. *Journal of the American Medical Informatics Association*. 78

[4] Alberti, C., Andor, D., Bogatyy, I., Collins, M., Gillick, D., Kong, L., Koo, T., Ma, J., Omernick, M., Petrov, S., Thanapirom, C., Tung, Z., and Weiss, D. (2017). Syntaxnet models for the conll 2017 shared task. *arXiv preprint arXiv:1703.04929*. 47

[5] Alsentzer, E., Murphy, J. R., Boag, W., Weng, W.-H., Jin, D., Naumann, T., and McDermott, M. B. A. (2019). Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323*. 115

[6] Arnold, T. and Fuller, H. J. A. (2016). Generic customizable simulation modules to emphasize error risk in medication order entry. *Proceedings of the International Symposium on Human Factors and Ergonomics in Health Care*, 5(1):150–153. 1

[7] Baker, S., Korhonen, A., and Pyysalo, S. (2016). Cancer hallmark text classification using convolutional neural networks. In *BioTxtM@COLING 2016*, pages 1–9. 2, 77

[8] Beltagy, I., Cohan, A., and Lo, K. (2019). Scibert: Pretrained contextualized embeddings for scientific text. *arXiv preprint arXiv:1903.10676*. 115

[9] Bernstein, B. and Potts, C. (2017). Optimizing the hyperparameter of which hyperparameter optimizer to use. 31

[10] Birkhead, G. S., Klompas, M., and Shah, N. R. (2015). Uses of electronic health records for public health surveillance to advance public health. *Annual Review of Public Health*, 36(1):345–359. 76

[11] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*. 4

[12] Bowman, S. E. (2013). Impact of electronic health record systems on information integrity: Quality and safety implications. *Perspectives in health information management / AHIMA, American Health Information Management Association*, 10. 1

[13] Camacho-Collados, J. and Pilehvar, M. T. (2018). From word to sense embeddings: A survey on vector representations of meaning. *arXiv preprint arXiv:1805.04032*. 49

[14] Carrell, D. S., Halgrim, S., Tran, D. T., Buist, D. S. M., Chubak, J., Chapman, W. W., and Savova, G. (2014). Using natural language processing to improve efficiency of manual chart abstraction in research: the case of breast cancer recurrence. *Am J Epidemiol*, 179(6):749–758. 9, 25, 48

[15] Che, Z., Cheng, Y., Sun, Z., and Liu, Y. (2017a). Exploiting convolutional neural network for risk prediction with medical feature embedding. *arXiv preprint arXiv:1701.07474*. 2, 77

[16] Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y. (2017b). Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8(1):6085. 6, 76

[17] Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. (2013). One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*. 113

[18] Cheng, Y., Wang, F., Zhang, P., and Hu, J. (2016). Risk prediction with electronic health records: A deep learning approach. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 432–440. 2, 77

[19] Chiticariu, L., Li, Y., and Reiss, F. R. (2013). Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 827–832. 4, 103

[20] Chorowski, J., Bahdanau, D., Serdyuk, D., et al. (2015). Attention-based models for speech recognition. In *Adv Neural Inf Process Syst*, pages 577–585. 41

[21] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*. 6, 24, 81

[22] Ciampi, M., Sicuranza, M., Esposito, A., Guarasci, R., and Pietro, G. D. (2016). A technological framework for ehr interoperability: experiences from italy. In *International Conference on Information and Communication Technologies for Ageing Well and e-Health*, pages 80–99. 1

[23] Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (elus). *international conference on learning representations*. 83

[24] Coden, A., Savova, G., Sominsky, I., Tanenblatt, M., Masanz, J., Schuler, K., Cooper, J., Guan, W., and De Groen, P. C. (2009). Automatically extracting cancer disease characteristics from pathology reports into a disease knowledge representation model. *Journal of biomedical informatics*, 42(5):937–949. 9, 48

[25] Conneau, A., Schwenk, H., Barrault, L., and Lecun, Y. (2017). Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1107–1116. 5, 49

[26] Connolly, J. L., Schnitt, S. J., Wang, H. H., Longtine, J. A., Dvorak, A., and Dvorak, H. F. (2003). Role of the surgical pathologist in the diagnosis and management of the cancer patient. 8

[27] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. 83

[28] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *north american chapter of the association for computational linguistics*. 5, 113

[29] DiCiccio, T. and Efron, B. (1996). Bootstrap confidence intervals. *Stat Sci*, pages 189–212. 35

[30] Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923. 62

[31] dos Santos, C. N., Xiang, B., and Zhou, B. (2015). Classifying relations by ranking with convolutional neural networks. pages 626–634. 64, 83

[32] dos Santos, C. N. and Zadrozny, B. (2014). Learning character-level representations for part-of-speech tagging. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1818–1826. 5

[33] Duque, A. B., Santos, L. L. J., Macdo, D., and Zanchettin, C. (2019). Squeezed very deep convolutional neural networks for text classification. *arXiv preprint arXiv:1901.09821*. 5

[34] Edgcomb, J. B. and Zima, B. (2019). Machine learning, natural language processing, and the electronic health record: Innovations in mental health services research. *Psychiatric Services*, 70(4):346–349. 2

[35] Edwards, B. K., Ward, E., Kohler, B. A., Eheman, C., Zauber, A. G., Anderson, R. N., Jemal, A., Schymura, M. J., Lansdorp-Vogelaar, I., Seeff, L. C., van Ballegooijen, M., Goede, S. L., and Ries, L. A. G. (2017). Annual report to the nation on the status of cancer, 1975-2014, featuring survival. *J Natl Cancer Inst*, 109(9). 48

[36] Efron, B. and Tibshirani, R. (1994). *An Introduction to the Bootstrap*. Chapman and Hall/CRC Monographs on Statistics and Applied Probability. Taylor and Francis. 90

[37] Erhan, D., Bengio, Y., Courville, A., et al. (2010). Why does unsupervised pre-training help deep learning? *J Mach Learn Res*, 11(Feb):625–660. 31, 42

[38] Ford, E., Carroll, J. A., Smith, H. E., Scott, D., and Cassell, J. A. (2016). Extracting information from the text of electronic medical records to improve case detection: a systematic review. *Journal of the American Medical Informatics Association*, 23(5):1007–1015. 2, 6

[39] Galassi, A., Lippi, M., and Torroni, P. (2019). Attention, please! a critical review of neural attention models in natural language processing. *arXiv preprint arXiv:1902.02181*. 5

[40] Gao, S., Ramanathan, A., and Tourassi, G. D. (2018a). Hierarchical convolutional attention networks for text classification. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 11–23. 53, 64, 65

[41] Gao, S., Young, M. T., Qiu, J. X., Yoon, H.-J., Christian, J. B., Fearn, P. A., Tourassi, G. D., and Ramanthan, A. (2018b). Hierarchical attention networks for information extraction from cancer pathology reports. *Journal of the American Medical Informatics Association*, 25(3):321–330. 50, 57, 59, 60, 111

[42] Garg, A. and Agarwal, M. (2019). Machine translation: A literature review. *arXiv preprint arXiv:1901.01122*. 4

[43] Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *Proc Int Conf Mach Learn*, pages 1243–1252. 64, 83

[44] Gehrmann, S., Dernoncourt, F., Li, Y., Carlson, E. T., Wu, J. T., Welt, J., Foote, J., Moseley, E. T., Grant, D. W., Tyler, P. D., and Celi, L. A. (2018). Comparing deep learning and concept extraction based methods for patient phenotyping from clinical narratives. *PLOS ONE*, 13(2). 6, 47, 76

[45] Grannis, S. J., Xu, H., Vest, J. R., Kasthurirathne, S. N., Bo, N., Moscovitch, B., Torkzadeh, R., and Rising, J. (2019). Evaluating the effect of data standardization and validation on patient matching accuracy. *Journal of the American Medical Informatics Association*, 26(5):447–456. 1

[46] Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*. 24

[47] Greenberg, N., Bansal, T., Verga, P., and McCallum, A. (2018). Marginal likelihood training of bilstm-crf for biomedical named entity recognition from disjoint label sets. In *EMNLP 2018: 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2824–2829. 6

[48] Greff, K., Srivastava, R., Koutník, J., et al. (2016). Lstm: A search space odyssey. *IEEE Trans Neural Netw Learn Syst*. 29

[49] He, B., Guan, Y., and Dai, R. (2019). Classifying medical relations in clinical text via convolutional neural networks. *Artificial Intelligence in Medicine*, 93:43–49. 2, 6, 77

[50] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780. 6, 24, 81

[51] Hong, N., Wen, A., Shen, F., Sohn, S., Liu, S., Liu, H., and Jiang, G. (2018). Integrating structured and unstructured ehr data using an fhir-based type system: A case study with medication data. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, 2017:74–83. 2

[52] Hu, M., Peng, Y., and Qiu, X. (2017). Reinforced mnemonic reader for machine comprehension. *Proc IJCAI*. 47

[53] Huang, J., Osorio, C., and Sy, L. W. (2018). An empirical evaluation of deep learning for icd-9 code assignment using mimic-iii clinical notes. *arXiv preprint arXiv:1802.02311*. 2, 6

[54] Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*. 84

[55] Hughes, M., Li, I., Kotoulas, S., and Suzumura, T. (2017). Medical text classification using convolutional neural networks. *arXiv preprint arXiv:1704.06841*. 2, 49

[56] Huh, M., Agrawal, P., and Efros, A. A. (2016). What makes imagenet good for transfer learning. *arXiv preprint arXiv:1608.08614*. 106

[57] Ishwari, K. S. D., Aneeze, A. K. R. R., Sudheesan, S., Karunaratne, H. J. D. A., Nugaliyadde, A., and Mallawarachchi, Y. (2019). Advances in natural language question answering: A review. *arXiv preprint arXiv:1904.05276*. 4

[58] Jagannatha, A. N. and Yu, H. (2016a). Bidirectional rnn for medical event detection in electronic health records. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 2016, pages 473–482. 2, 25, 49, 77, 82

[59] Jagannatha, A. N. and Yu, H. (2016b). Structured prediction models for rnn based sequence labeling in clinical text. In *Proc Conf Empir Methods Nat Lang Process*, volume 2016, pages 856–865. 49

[60] Jensen, K., Soguero-Ruiz, C., yvind Mikalsen, K., Lindsetmo, R.-O., Kouskoumvekaki, I., Girolami, M., Skrovseth, S. O., and Augestad, K. M. (2017). Analysis of free text in electronic health records for identification of cancer patient trajectories. *Scientific Reports*, 7(1):46226. 2, 6, 76

[61] Jin, D. and Szolovits, P. (2018). Hierarchical neural networks for sequential sentence classification in medical scientific abstracts. *empirical methods in natural language processing*, pages 3100–3109. 78

[62] Johnson, R. and Zhang, T. (2015). Semi-supervised convolutional neural networks for text categorization via region embedding. In *Adv Neural Inf Process Syst*, NIPS'15, pages 919–927, Cambridge, MA, USA. MIT Press. 42

[63] Johnson, R. and Zhang, T. (2016). Supervised and semi-supervised text categorization using lstm for region embeddings. In *Proc Int Conf Mach Learn*, ICML'16, pages 526–534. JMLR.org. 42

[64] Johnson, R. and Zhang, T. (2017). Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 562–570. 5

[65] Jouhet, V., Defossez, G., Burgun, A., et al. (2012). Automated classification of free-text pathology reports for registration of incident cases of cancer. *Methods Inf Med*, 51(3):242. 25

[66] Kalidindi, K. V. (2019). Deconstructing word embeddings. *arXiv preprint arXiv:1902.00551*. 4

[67] Kalra, S., Li, L., and Tizhoosh, H. R. (2019). Automatic classification of pathology reports using tf-idf features. *arXiv preprint arXiv:1903.07406*. 2, 6

[68] Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. 4, 6, 24, 59, 77, 81, 82, 87

[69] Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. *Proc ICLR*. 60, 89

[70] Koleck, T. A., Dreisbach, C. N., Bourne, P. E., and Bakken, S. (2019). Natural language processing of symptoms documented in free-text narratives of electronic health records: a systematic review. *Journal of the American Medical Informatics Association*, 26(4):364–379. 2

[71] Kumar, A., Irsoy, O., Ondruska, P., et al. (2016). Ask me anything: Dynamic memory networks for natural language processing. In *Proc Int Conf Mach Learn*, pages 1378–1387. 24

[72] Le, H. T., Cerisara, C., and Denis, A. (2018). Do convolutional networks need to be deep for text classification. *Proc Conf AAAI Artif Intell*, pages 29–36. 49

[73] Lee, J., Song, H.-J., Yoon, E., Park, S.-B., Park, S.-H., Seo, J.-W., Park, P., and Choi, J. (2018). Automated extraction of biomarker information from pathology reports. *BMC medical informatics and decision making*, 18(1):29. 9, 48

[74] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2019). Biobert: a pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*. 115

[75] Lee, J. Y. and Dernoncourt, F. (2016). Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–520. 78

[76] Lei, T., Zhang, Y., Wang, S. I., Dai, H., and Artzi, Y. (2018). Training rnns as fast as cnns. In *EMNLP 2018: 2018 Conference on Empirical Methods in Natural Language Processing*. 5

[77] Letarte, G., Paradis, F., Gigure, P., and Laviolette, F. (2018). Importance of self-attention for sentiment analysis. In *BlackboxNLP@EMNLP*, pages 267–275. 5

[78] Li, J., Luong, M.-L., and Jurafsky, D. (2015). A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1106–1115. 31

[79] Li, P. and Huang, H. (2016). Clinical information extraction via convolutional neural network. *arXiv preprint arXiv:1603.09381*. 25

[80] Li, Y. and Martinez, D. (2010). Information extraction of multiple categories from pathology reports. *Proc ALTA*, pages 41–48. 9, 49

[81] Liang, D., Xu, W., and Zhao, Y. (2017). Combining word-level and character-level representations for relation classification of informal text. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 43–47. 5

[82] Lipton, Z., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*. 24, 29, 81

[83] Lipton, Z. C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*. 5

[84] Lipton, Z. C., Kale, D. C., Elkan, C., and Wetzell, R. (2016). Learning to diagnose with lstm recurrent neural networks. *international conference on learning representations*. 77

[85] Liu, J., Zhang, Z., and Razavian, N. (2018). Deep ehr: Chronic disease prediction using medical notes. *arXiv preprint arXiv:1808.04928*, pages 440–464. 2, 6

[86] Lowy, D. and Collins, F. (2016). Aiming high—changing the trajectory for cancer. *New England Journal of Medicine*, 374(20):1901–1904. 22

[87] Martin, E. and Cundy, C. (2018). Parallelizing linear recurrent neural nets over sequence length. *international conference on learning representations*. 5

[88] Martinez, D. and Li, Y. (2011). Information extraction from pathology reports in a hospital setting. In *Proc ACM Int Conf Inf Knowl Manag*, pages 1877–1882. ACM. 9, 25, 49

[89] Mays, J. A. and Mathias, P. C. (2019). Measuring the rate of manual transcription error in outpatient point-of-care testing. *Journal of the American Medical Informatics Association*, 26(3):269–272. 1

[90] McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017). Learned in translation: Contextualized word vectors. *arXiv preprint arXiv:1708.00107*. 113

[91] McCann, B., Keskar, N. S., Xiong, C., and Socher, R. (2018). The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*. 6

[92] Menachemi, N. and Collum, T. H. (2011). Benefits and drawbacks of electronic health record systems. *Risk Management and Healthcare Policy*, 4:47–55. 76, 105

[93] Mikolov, T., Chen, K., Corrado, G., et al. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. 28

[94] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Adv Neural Inf Process Syst*, pages 3111–3119. 28, 60

[95] Mork, J. G., Jimeno-Yepes, A., and Aronson, A. R. (2013). The nlm medical text indexer system for indexing biomedical literature. In *BioASQ@ CLEF*. 1

[96] Mullenbach, J., Wiegreffe, S., Duke, J., Sun, J., and Eisenstein, J. (2018). Explainable prediction of medical codes from clinical text. In *NAACL HLT 2018: 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 1101–1111. 6, 47, 77

[97] Mntyl, M. V., Graziotin, D., and Kuutila, M. (2018). The evolution of sentiment analysisa review of research topics, venues, and top cited papers. *Computer Science Review*, 27:16–32. 4

[98] National Cancer Institute (2016). Coding guidelines breast c500 -c509. 26

[99] National Cancer Institute (2017). Overview of the seer program. 23

[100] Nguyen, A. N., Lawley, M. J., Hansen, D. P., Bowman, R. V., Clarke, B. E., Duhig, E. E., and Colquist, S. (2010). Symbolic rule-based classification of lung cancer stages from free-text pathology reports. *J Am Med Inform Assoc*, 17(4):440–445. 9, 48

[101] Nguyen, A. N., Moore, J., ODwyer, J., and Philpot, S. (2015). Assessing the utility of automatic cancer registry notifications data extraction from free-text pathology reports. In *AMIA Annual Symposium Proceedings*, volume 2015, page 953. American Medical Informatics Association. 9, 48

[102] Nguyen, P., Tran, T., Wickramasinghe, N., and Venkatesh, S. (2017). Deepr: A convolutional net for medical records. *IEEE Journal of Biomedical and Health Informatics*, 21(1):22–30. 2, 77

[103] Ning, H.-C., Lin, C.-N., Chiu, D. T.-Y., Chang, Y.-T., Wen, C.-N., Peng, S.-Y., Chu, T.-L., Yu, H.-M., and Wu, T.-L. (2016). Reduction in hospital-wide clinical laboratory specimen identification errors following process interventions: A 10-year retrospective observational study. *PLOS ONE*, 11(8). 1

[104] O'Connor, J. D. (2015). Reducing post analytical error: Perspectives on new formats for the blood sciences pathology report. *The Clinical biochemist. Reviews / Australian Association of Clinical Biochemists*, 36(1):7–20. 1

[105] Otter, D. W., Medina, J. R., and Kalita, J. K. (2018a). A survey of the usages of deep learning in natural language processing. *arXiv preprint arXiv:1807.10854*. 2

[106] Otter, D. W., Medina, J. R., and Kalita, J. K. (2018b). A survey of the usages of deep learning in natural language processing. *arXiv preprint arXiv:1807.10854*. 76

[107] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *Proc Int Conf Mach Learn*, pages 1310–1318. 50

[108] Pathak, J., Kho, A. N., and Denny, J. C. (2013). Electronic health records-driven phenotyping: challenges, recent advances, and perspectives. *Journal of the American Medical Informatics Association*, 20. 6, 76

[109] Peng, H., Mou, L., Li, G., Chen, Y., Lu, Y., and Jin, Z. (2015). A comparative study on regularization strategies for embedding-based neural networks. In *Proc Conf Empir Methods Nat Lang Process*, pages 2106–2111. 58

[110] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proc Conf Empir Methods Nat Lang Process*, volume 14, pages 1532–1543. 28

[111] Peters, M. E., Ammar, W., Bhagavatula, C., and Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1756–1765. 113

[112] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. S. (2018). Deep contextualized word representations. In *NAACL HLT 2018: 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 2227–2237. 5, 113

[113] Pineda, A. L., Walk, O. J. B. D., Venkataraman, G. R., Zehnder, A. M., Ayyar, S., Page, R. L., Bustamante, C. D., and Rivas, M. A. (2019). Fastag: automatic text classification of unstructured medical narratives. *bioRxiv*, page 429720. 2, 6

[114] Pinto, A. M., Oliveira, H. G., and Alves, A. O. (2016). Comparing the performance of different nlp toolkits in formal and social media text. In *Proc SLATE*, volume 51, page 16. 9, 49

[115] Powsner, S., Costa, J., and Homer, R. (2000). Clinicians are from mars and pathologists are from venus: clinician interpretation of pathology reports. *Arch Pathol Lab Med*, 124(7):1040–1046. 42

[116] Qiu, J. X., Yoon, H.-J., Fearn, P. A., and Tourassi, G. D. (2018a). Deep learning for automated extraction of primary sites from cancer pathology reports. *IEEE Journal of Biomedical and Health Informatics*, 22(1):244–251. 9, 11, 49, 59, 78, 87

[117] Qiu, J. X., Yoon, H.-J., Fearn, P. A., and Tourassi, G. D. (2018b). Deep learning for automated extraction of primary sites from cancer pathology reports. *IEEE J Biomed Health Inform*, 22(1):244–251. 25, 34

[118] Qiu, J. X., Yoon, H.-J., Srivastava, K., Watson, T. P., Christian, J. B., Ramanathan, A., Wu, X. C., Fearn, P. A., and Tourassi, G. D. (2018). Scalable deep text comprehension for cancer surveillance on high-performance computing. *BMC bioinformatics*, 19(18):488. 78, 87

[119] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2018). Language models are unsupervised multitask learners. 5

[120] Rahul, P. V. S. S., Sahu, S. K., and Anand, A. (2017). Biomedical event trigger identification using bidirectional recurrent neural network based models. In *BioNLP 2017*, pages 316–321. 82

[121] Rajkomar, A., Oren, E., Chen, K., Dai, A. M., Hajaj, N., Hardt, M., Liu, P. J., Liu, X., Marcus, J., Sun, M., Sundberg, P., Yee, H., Zhang, K., Zhang, Y., Flores, G.,

Duggan, G. E., Irvine, J., Le, Q., Litsch, K., Mossin, A., Tansuwan, J., Wang, D., Wexler, J., Wilson, J., Ludwig, D., Volchenboum, S. L., Chou, K., Pearson, M., Madabushi, S., Shah, N. H., Butte, A. J., Howell, M. D., Cui, C., Corrado, G. S., and Dean, J. (2018). Scalable and accurate deep learning with electronic health records. *arXiv preprint arXiv:1801.07860*, 1(1):18. 2, 6, 105

[122] Retegui, R. and Ratt, S. (2018). Comparison of metamap and ctakes for entity extraction in clinical notes. *BMC Medical Informatics and Decision Making*, 18(3):74. 6

[123] Rios, A. and Kavuluru, R. (2015). Convolutional neural networks for biomedical text classification: application in indexing biomedical articles. In *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 258–267. 2

[124] Romanov, A. and Shivade, C. (2018). Lessons from natural language inference in the clinical domain. *empirical methods in natural language processing*, pages 1586–1596. 6

[125] Salehinejad, H., Baarbe, J., Sankar, S., Barfett, J., Colak, E., and Valaee, S. (2018). Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*. 81

[126] Sankhavara, J. (2018). Biomedical document retrieval for clinical decision support system. In *Proceedings of ACL 2018, Student Research Workshop*, pages 84–90. 6, 76

[127] Schreiber, R., Sittig, D. F., Ash, J. S., and Wright, A. (2017). Orders on file but no labs drawn: Investigation of machine and human errors caused by an interface idiosyncrasy. *Journal of the American Medical Informatics Association*, 24(5):958–963. 1

[128] Senel, L. K., Utlu, I., Yucesoy, V., Koc, A., and Cukur, T. (2018). Semantic structure and interpretability of word embeddings. *IEEE Trans Audio Speech Lang Process*, 26(10):1769–1779. 56

[129] Shavit, A. and Griesemer, J. (2011). Transforming objects into data: How minute technicalities of recording species location entrench a basic challenge for biodiversity. pages 169–193. 1

[130] Shen, Z., Zhang, M., Yi, S., Yan, J., and Zhao, H. (2018). Factorized attention: Self-attention with linear complexities. *arXiv: Computer Vision and Pattern Recognition*. 81

[131] Shickel, B., Tighe, P. J., Bihorac, A., and Rashidi, P. (2018). Deep ehr: A survey of recent advances in deep learning techniques for electronic health record (ehr) analysis. *IEEE Journal of Biomedical and Health Informatics*, 22(5):1589–1604. 2, 6, 76

[132] Siegel, R. L., Miller, K. D., and Jemal, A. (2019). Cancer statistics, 2019. *CA: A Cancer Journal for Clinicians*, 69(1):7–34. 2

[133] Singh, A., Nadkarni, G., Gottesman, O., Ellis, S. B., Bottinger, E. P., and Guttag, J. V. (2015). Incorporating temporal ehr data in predictive models for risk stratification of renal function deterioration. *Journal of Biomedical Informatics*, 53:220–228. 76

[134] Singh, H., Meyer, A. N. D., and Thomas, E. J. (2014). The frequency of diagnostic errors in outpatient care: estimations from three large observational studies involving us adult populations. *BMJ Quality Safety*, 23(9):727–731. 1

[135] Sitapati, A., Kim, H., Berkovich, B., Marmor, R., Singh, S., El-Kareh, R., Clay, B., and Ohno-Machado, L. (2017). Integrated precision medicine: the role of electronic health records in delivering personalized treatment. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 9(3). 105

[136] Soutner, D. and Mller, L. (2013). Application of lstm neural networks in language modelling. In *International Conference on Text, Speech and Dialogue*, pages 105–112. 5

[137] Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. In *ACL 2019 : The 57th Annual Meeting of the Association for Computational Linguistics*. 104

[138] Tang, G., Mller, M., Gonzales, A. R., and Sennrich, R. (2018). Why self-attention? a targeted evaluation of neural machine translation architectures. *empirical methods in natural language processing*, pages 4263–4272. 5, 83

[139] Trinh, T. H., Dai, A. M., Luong, M.-T., and Le, Q. V. (2018). Learning longer-term dependencies in rnns with auxiliary losses. *arXiv: Learning*. 12

[140] van der Maaten, L. and Hinton, G. E. (2008). Visualizing data using t-sne. *J Mach Learn Res*, 9:2579–2605. 68

[141] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *neural information processing systems*, pages 5998–6008. 5, 14, 50, 55, 56, 83, 84

[142] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019). Glue: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR 2019 : 7th International Conference on Learning Representations*. 6

[143] Wang, W., Yan, M., and Wu, C. (2018a). Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. *meeting of the association for computational linguistics*, 1:1705–1714. 5

[144] Wang, Y., Wang, L., Rastegar-Mojarad, M., Moon, S., Shen, F., Afzal, N., Liu, S., Zeng, Y., Mehrabi, S., Sohn, S., and Liu, H. D. (2018b). Clinical information extraction applications: A literature review. *Journal of Biomedical Informatics*, 77:34–49. 2, 9, 49, 103

[145] Weegar, R. and Dalianis, H. (2015). Creating a rule based system for text mining of norwegian breast cancer pathology reports. In *Proc LOUHI*, pages 73–78. 9, 48

[146] Wu, Y., Jiang, M., Xu, J., Zhi, D., and Xu, H. (2017). Clinical named entity recognition using deep learning models. In *AMIA ... Annual Symposium proceedings. AMIA Symposium*, volume 2017, pages 1812–1819. 2

[147] Xiao, C., Choi, E., and Sun, J. (2018). Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. *Journal of the American Medical Informatics Association*, 25(10):1419–1428. 49, 76, 103, 105

[148] Xie, F., Lee, J., Munoz-Plaza, C. E., Hahn, E. E., and Chen, W. (2017). Application of text information extraction system for real-time cancer case identification in an integrated healthcare organization. *Journal of pathology informatics*, 8. 9, 48

[149] Xiong, C., Merity, S., and Socher, R. (2016). Dynamic memory networks for visual and textual question answering. *international conference on machine learning*, pages 2397–2406. 5

[150] Yala, A., Barzilay, R., Salama, L., Griffin, M., Sollender, G., Bardia, A., Lehman, C., Buckley, J. M., Coopey, S. B., Polubriaginof, F., et al. (2017). Using machine learning to parse breast pathology reports. *Breast cancer research and treatment*, 161(2):203–211. 9, 49

[151] Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS 2019 : Thirty-third Conference on Neural Information Processing Systems*. 114

[152] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A. J., and Hovy, E. H. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. 12, 18, 25, 28, 29, 43, 50, 59

[153] Yao, L., Mao, C., and Luo, Y. (2018). Clinical text classification with rule-based features and knowledge-guided convolutional neural networks. In *2018 IEEE International Conference on Healthcare Informatics Workshop (ICHI-W)*, pages 70–71. 6

[154] Yim, W.-w., Denman, T., Kwan, S. W., and Yetisgen, M. (2016). Tumor information extraction in radiology reports for hepatocellular carcinoma patients. *AMIA Summits on Translational Science Proceedings*, 2016:455. 9, 49

[155] Yoon, H., Ramanathan, A., and Tourassi, G. (2016). Multi-task deep neural networks for automated extraction of primary site and laterality information from cancer pathology reports. In *INNS Conference on Big Data*, pages 195–204. Springer. 43

[156] Young, T., Hazarika, D., Poria, S., and Cambria, E. (2017). Recent trends in deep learning based natural language processing. *arXiv preprint arXiv:1708.02709*. 2, 4, 49

[157] Yu, A. W., Dohan, D., Luong, M.-T., Zhao, R., Chen, K., Norouzi, M., and Le, Q. V. (2018). Qanet: Combining local convolution with global self-attention for reading comprehension. *international conference on learning representations*. 5, 50, 83

[158] Zhang, L., Wang, S., and Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdiscip Rev Data Min Knowl Discov*, 8(4). 47

[159] Zhang, S., Wu, Y., Che, T., Lin, Z., Memisevic, R., Salakhutdinov, R. R., and Bengio, Y. (2016). Architectural complexity measures of recurrent neural networks. *neural information processing systems*, pages 1830–1838. 81

[160] Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. *arXiv preprint arXiv:1509.01626*. 4, 9, 49

[161] Zhao, J., Papapetrou, P., Asker, L., and Bostrm, H. (2017). Learning from heterogeneous temporal data in electronic health records. *Journal of Biomedical Informatics*, 65:105–119. 76

[162] Zheng, S., Lu, J. J., Appin, C., Brat, D., and Wang, F. (2015). Support patient search on pathology reports with interactive online learning based data extraction. *Journal of pathology informatics*, 6. 9, 49

[163] Zhu, Q., Li, X., Conesa, A., and Pereira, C. (2018). Gram-cnn: a deep learning approach with local context for named entity recognition in biomedical text. *Bioinformatics*, 34(9):1547–1554. 6

Appendices

A Chapter 2 Supporting Information

This appendix contains the supporting information for the paper "Hierarchical attention networks for information extraction from cancer pathology reports."

A.1 Preprocessing Pipeline

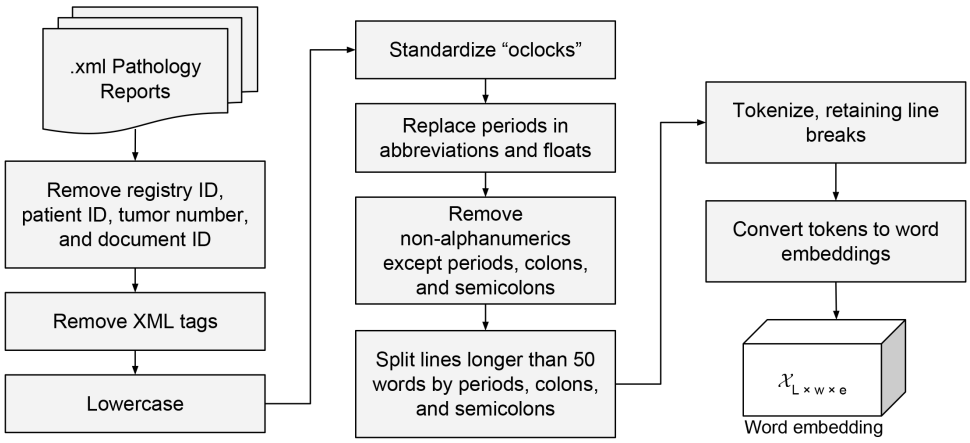


Figure A.1: Preprocessing pipeline for our cancer pathology report dataset

A.2 Preprocessing Procedures

1. Remove identifier segments (registry ID, patient ID, tumor number, and document ID)
2. Remove XML tags
3. Lowercase
4. Replace line breaks and tabs with spaces
5. Standardize oclocks by replacing all references to clock time with a number 1-12 followed by the string oclock
6. Remove periods in the abbreviations dr., am., and pm. (this allows splitting lines by periods later)

7. Remove periods in floats by replacing all instances of floats with the string floattoken (this allows splitting lines by periods later)

8. Replace names in pathology reports (represented in the format **NAME[____]) with the string nametoken

9. Replace dates in pathology reports (represented in the format **DATE[____]) with the string datetoken

10. Replace all integers higher than 100 with the string largeint (to reduce the number of unique tokens associated with numbers)

11. Remove non-alphanumerics except periods, colons, and semicolons

12. Split lines longer than 50 words by periods, colons, and semicolons

13. Tokenize document, retaining line breaks

14. Replace each token with the appropriate Word2Vec or Glove embedding

A.3 Corpus Histogram

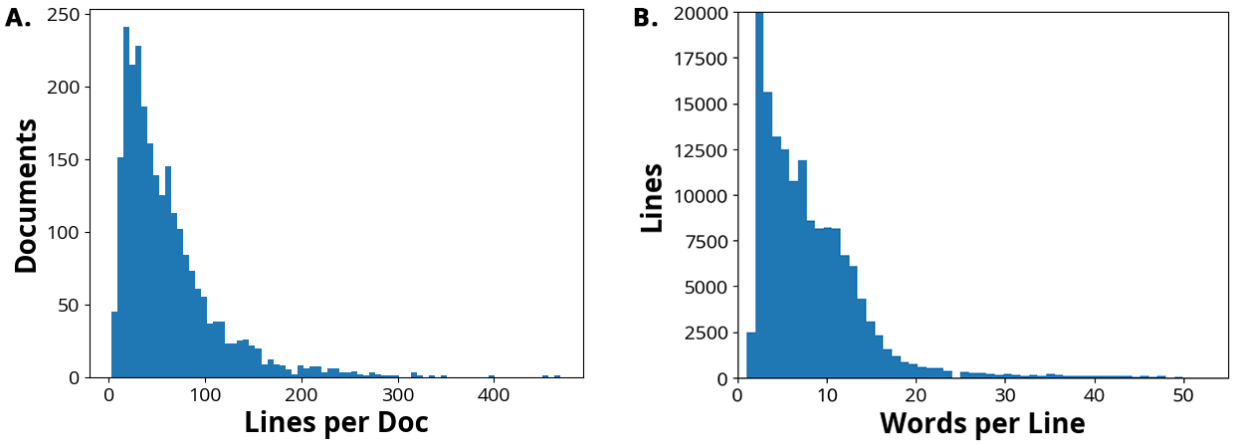


Figure A.2: (A.) Average number of lines per report and (B.) average number of words per line across all 2505 pathology reports in corpus

A.4 LSTMs and GRUs

LSTMs are a type of recurrent unit that use gating functions and an internal memory to control how input at each time step is processed and saved[3]. LSTMs utilize three gating functions that make decisions based on the information h_{t-1} stored in internal memory and the information x_t at the current timestep. The "forget gate" outputs a value between 0 and 1 that determines how much of the internal memory from the previous timestep to retain (Eqn. A.1).

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{A.1}$$

The "candidate value" is the processed input from the current timestep (Eqn. A.2), and the "input gate" controls how much of the candidate value is saved into the internal memory (Eqn. A.3).

$$cand_t = \tanh(W_{cand}[h_{t-1}, x_t] + b_{cand}) \tag{A.2}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{A.3}$$

The new state of the LSTM's internal memory then becomes the sum of the previous memory state multiplied by the forget gate and the candidate value multiplied by the input gate (Eqn. A.4).

$$C_t = f_t * C_{t-1} + i_t * cand_t \tag{A.4}$$

Finally, an "output gate" controls how much of the internal memory to output at the current timestep (Eqn. A.5). The outputted memory goes through a final tanh activation before passing to the next layer (Eqn. A.6).

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{A.5}$$

$$h_t = o_t * \tanh(C_t) \tag{A.6}$$

[3]Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):17351780

GRUs are an alternative to LSTMs. Like LSTMs, GRUs also use gating functions to control the flow of information into and out of the unit. However, GRUs only use two gates instead of three. An "update gate" determines how much of the previous memory to keep (Eqn. A.7), and a "reset gate" determines how to combine the input at the current timestep with the previous memory (Eqn. A.8 and A.9). Another difference from LSTMs is that GRUs output their entire internal state without an additional gating function or activation function (Eqn. 10). GRUs have been shown to perform just as well as LSTMs for many tasks and can be faster to train because they have fewer parameters[4].

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z) \tag{A.7}$$

$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r) \tag{A.8}$$

$$cand_t = \tanh(W_{cand}(r * [h_{t-1}, x_t]) + b_{cand}) \tag{A.9}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * cand_t \tag{A.10}$$

A.5 Attention Mechanism

For our attention mechanism, we use a single-hidden layer neural network with a softmax output. The attention mechanism inputs are the recurrent layer outputs at each timestep, which are fed into a tanh activation in the hidden layer (Eqn. A.11). The attention mechanism softmax output corresponds to how to weigh the output at each timestep (Eqn. A.12). We use the dot product of the attention softmax ouput and the recurrent layer output as the final attention-weighted recurrent output (Eqn. A.13).

$$u_{it} = \tanh(W_{att}h_{it} + b_{att}) \tag{A.11}$$

$$a_{it} = \frac{\exp(W_{softmax}u_{it} + b_{softmax})}{\sum_t \exp(W_{softmax}u_{it} + b_{softmax})} \tag{A.12}$$

$$emb_i = \sum_t a_{it}h_{it} \tag{A.13}$$

[4]Chung J, Gulcehre C, Cho K, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:14123555. 2014.

A.6 Hyperparameter Optimization Setup

1. Hyperparameter optimization performed using Python sci-kit optimize (skopt) package
2. We use sequential optimization using gradient boosted trees
3. For each of the two classification tasks (primary site and histological grade), we first perform an 0.8/0.1/0.1 train/validation/test split on the corresponding dataset
4. For each classification task, we then run hyperparameter optimization for 50 iterations with 10 random restarts
5. For each iteration, we train a new HAN for 20 epochs on the train set
6. To account for instability in model performance during training, we use the average model performance on the validation set in epochs 11-20 to evaluate the performance of each hyperparameter configuration
7. We save the best hyperparameter configuration for each classification task
8. The test set is not used during hyperparameter optimization

Table A.1: Optimized hyperparameters for HAN for each classification task, discovered through gradient boosted tree-based search. We note that for the hyperparameters Number of RNN units per hierarchy and Size of attention hidden layer, the search space was not expanded above 200 and 300 respectively because in both cases the best performing hyperparameter was below the upper limit and was simply rounded up to the nearest 50.

Hyperparameter	**Primary Site**	**Histology**	**Search Space**
Type of word embedding	GloVe	GloVe	(Word2Vec, Glove)
Size of word embeddings (rounded to nearest 50)	350	350	(100 - 500)
Type of RNN unit	GRU	LSTM	(GRU, LSTM)
Number of RNN units per hierarchy (rounded to nearest 50)	200	200	(50 200)
Size of attention hidden layer (rounded to nearest 50)	300	150	(50 300)
Dropout keep rate (rounded to nearest 0.1)	0.5	0.5	(0.5 or None)

A.7 TF-IDF Weighted Word Embeddings

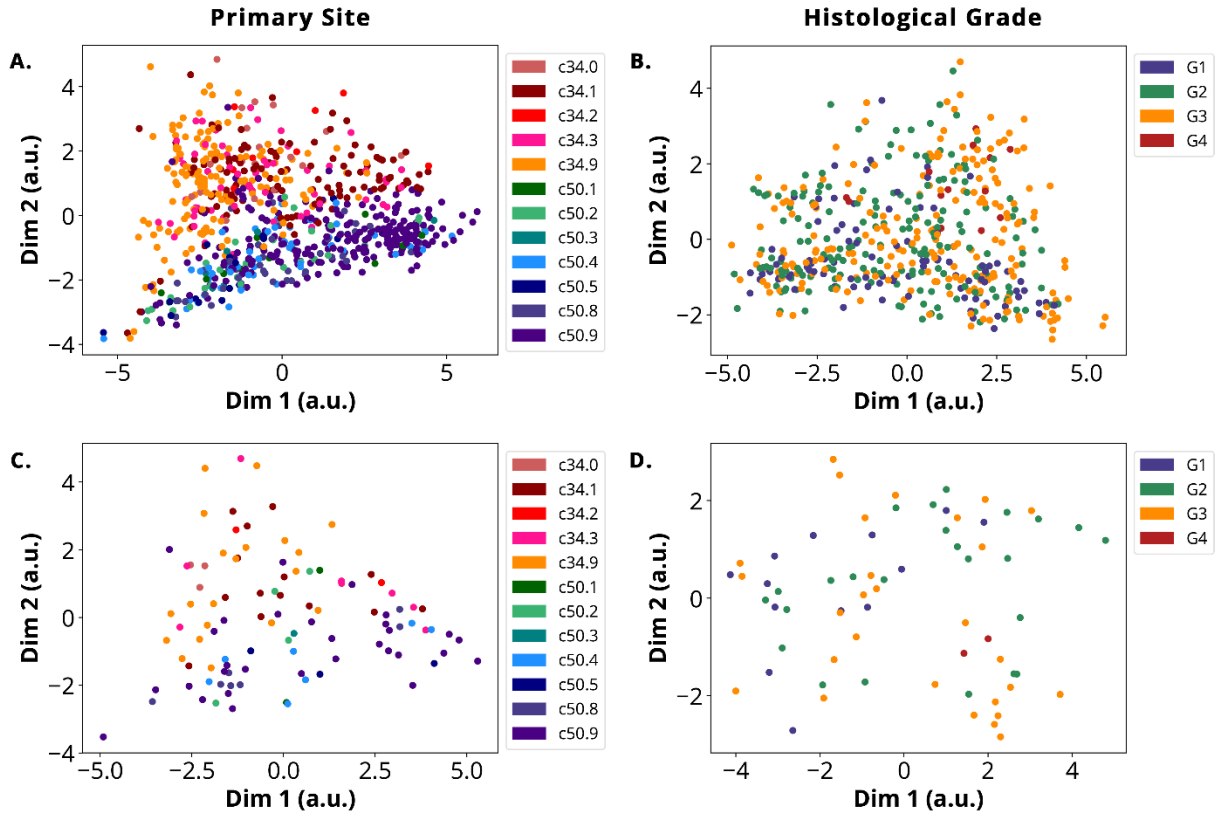


Figure A.3: TF-IDF weighted word embeddings reduced to 2-dimensions via PCA for (A.) primary site train reports, (B.) histological grade train reports, (C.) primary site test reports, and (D.) histological grade test reports

A.8 Bootstrapping Procedures for Metrics

1. Run 10-fold cross validation on dataset, save predicted test set labels for each fold
2. Concatenate all predicted test set labels (hereon referred to as original predictions)
3. Bootstrap (with replacement) from original predictions to create a new set of predicted labels of the same size (hereon referred to as bootstrapped set)
4. Calculate micro and macro F-score on bootstrapped set
5. Repeat steps (3) and (4) 2000 times, saving the scores each time

6. Calculate the mean for each metric by using the mean value across all 2000 runs

7. Calculate the 95% confidence interval for each metric by finding the 2.5 and 97.5 percentile entry for that metric within the 2000 runs (since F-score is not normally distributed)

A.9 Doc2Vec Embeddings

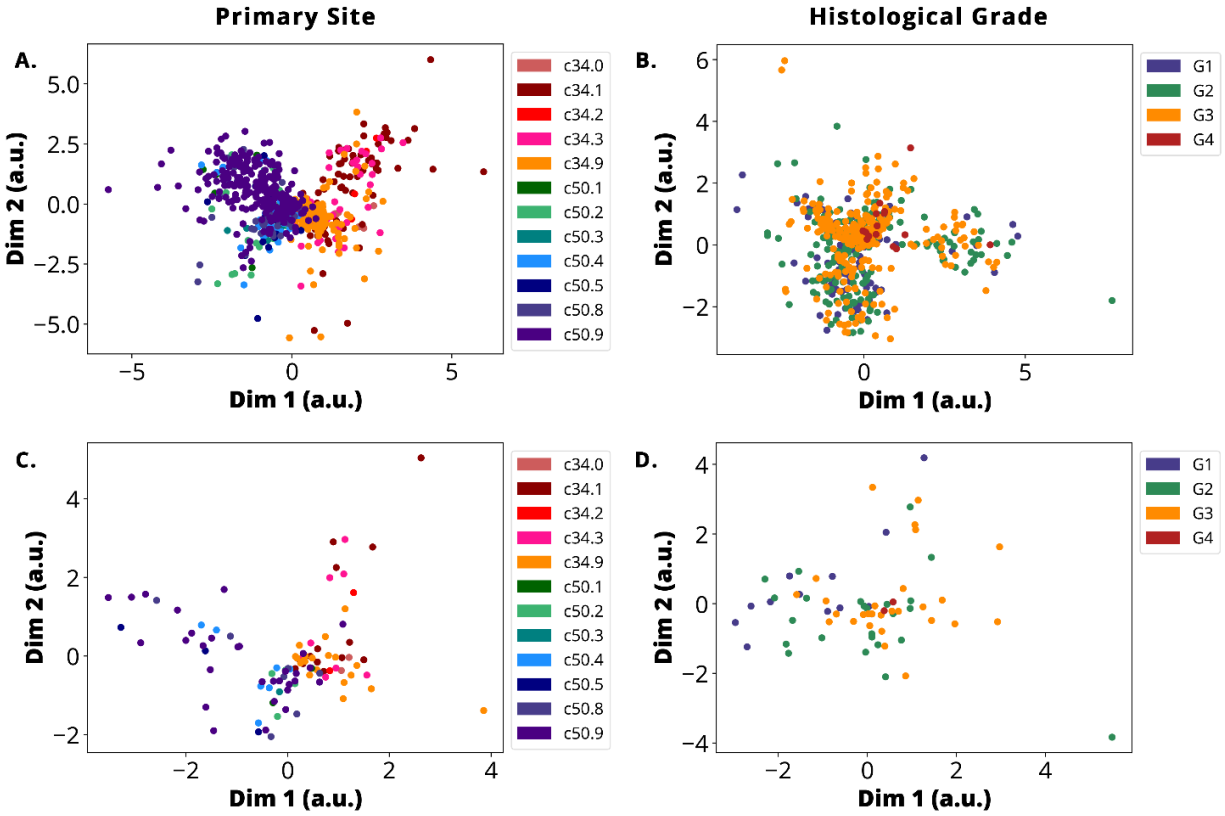


Figure A.4: Doc2Vec embeddings reduced to 2-dimensions via PCA for (A.) primary site train reports, (B.) histological grade train reports, (C.) primary site test reports, and (D.) histological grade test reports

A.10 CNN Embeddings

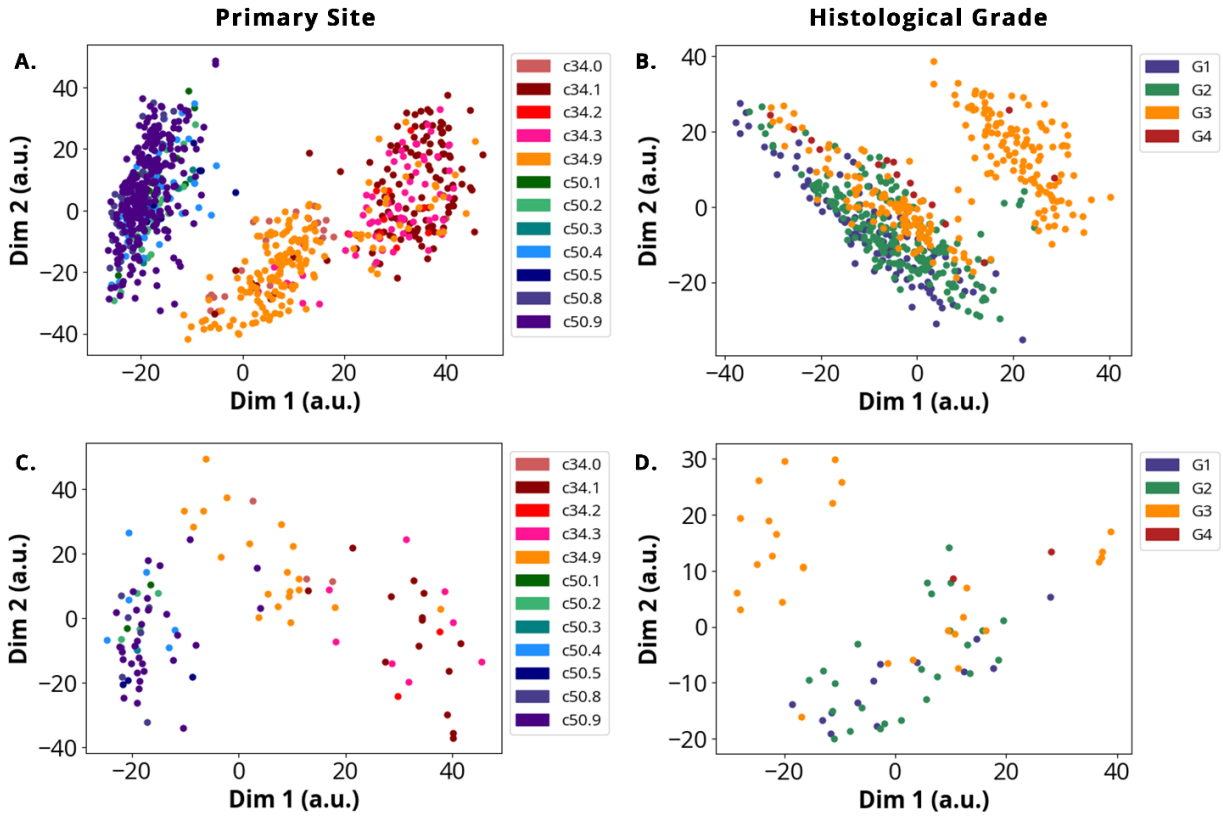


Figure A.5: CNN document embeddings reduced to 2-dimensions via PCA for (A.) primary site train reports, (B.) histological grade train reports, (C.) primary site test reports, and (D.) histological grade test reports

A.11 Most Important Words for Each Classification Task

Table A.2: The most important words for each classification task, based on the average word-level attention scores assigned to each word

Primary Site			**(Word2Vec, Glove)**		
mainstem	carina	buttock	poorly	undifferentiated	moderately
adenoca	cusa	temporal	g2	g3	intermediate
lul	arm	upper	high	ii	well
lower	middle	retro	iii	i	arising
breast	rul	sputum	dlr	g1	2

A.12 Model Prediction Time

Table A.3: The average time (in ms) required to classify a single pathology report for each trained classifier. All models were tested using on a single Intel Core i7-3770 CPU @ 3.40GHz. We note that HANs train faster than regular RNNs because whereas RNNs read the entire document as one long sequence, HANS break documents down into shorter, more manageable sequences that are faster to process.

Classifier	Primary Site	Histological Grade
Naive Bayes	0.005	0.002
Logistic Regression	0.005	0.003
Support Vector Machine	1.345	1.025
Random Forest	0.681	0.801
XGBoost	0.210	0.221
Recurrent Neural Network w/o Attention	67.204	77.465
Recurrent Neural Network w/ Attention	76.275	85.915
Convolutional Neural Network	24.968	28.381
Hierarchical Attention Network	53.379	63.250

A.13 Confusion Matrices

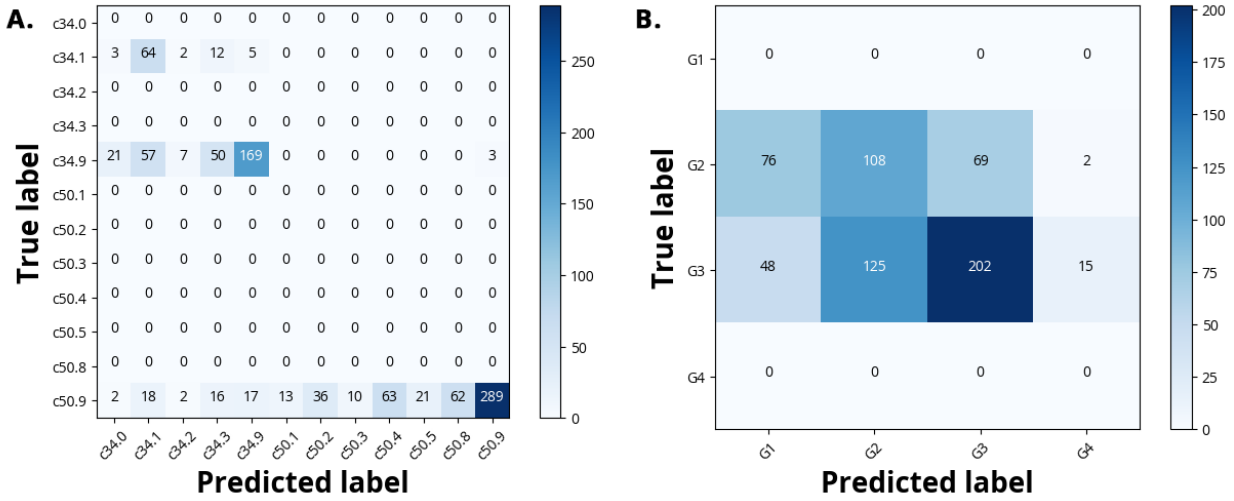


Figure A.6: Naive Bayes Confusion Matrix for (A.) Primary Site and (B.) Histological Grade

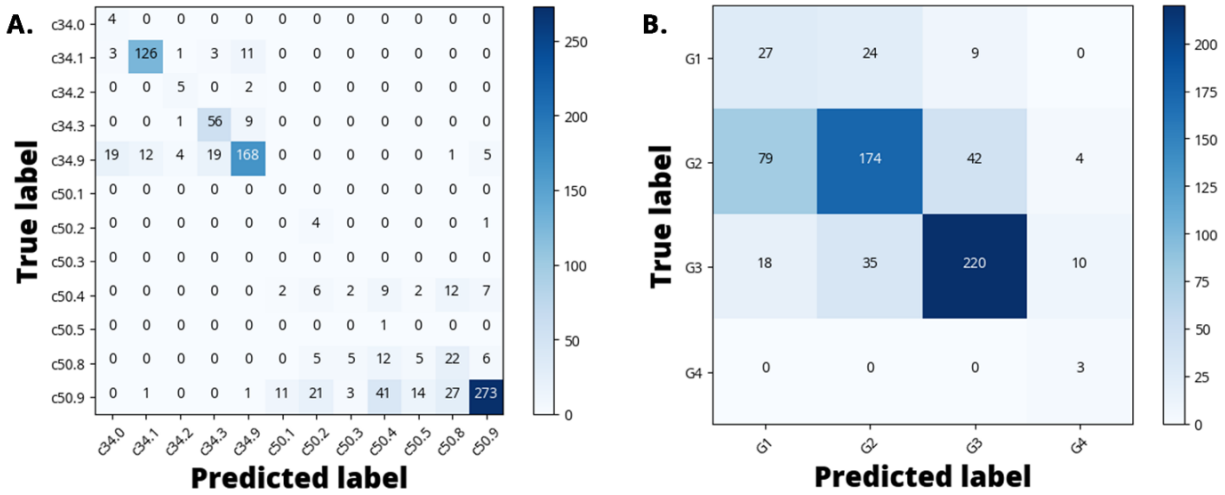


Figure A.7: Logistic Regression Confusion Matrix for (A.) Primary Site and (B.) Histological Grade

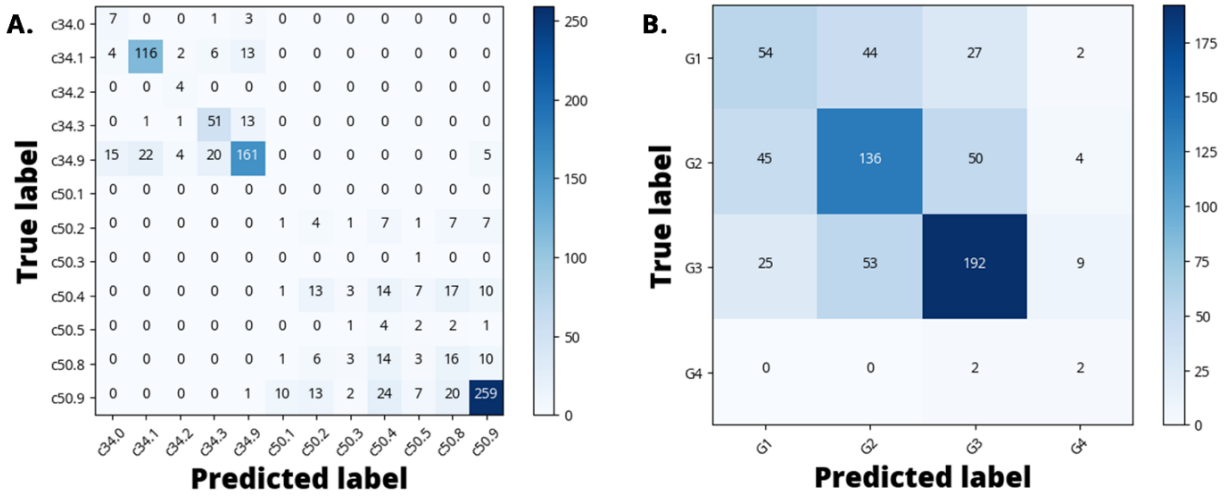


Figure A.8: Support Vector Machine Confusion Matrix for (A.) Primary Site and (B.) Histological Grade

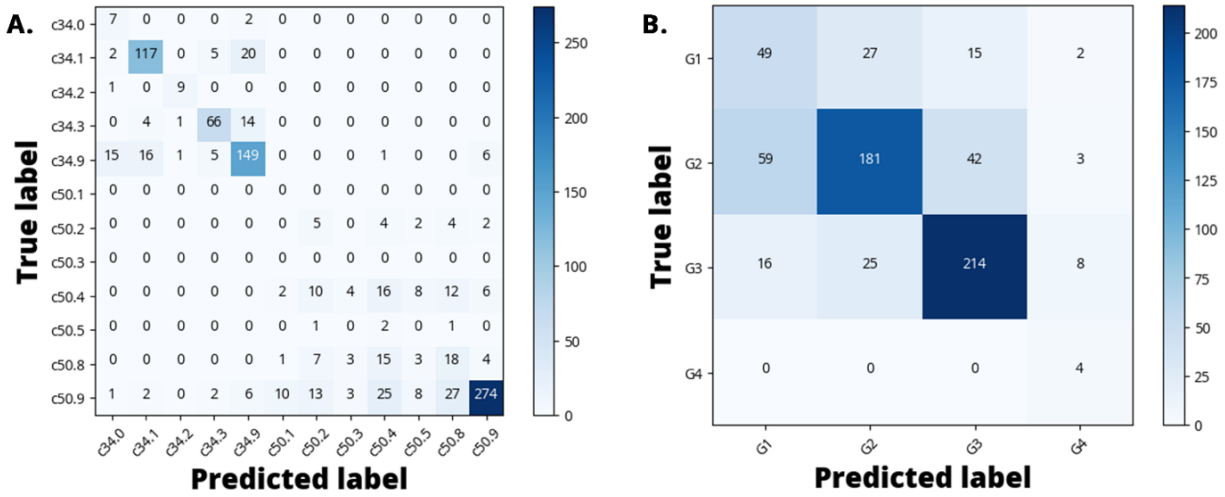


Figure A.9: Random Forest Machine Confusion Matrix for (A.) Primary Site and (B.) Histological Grade

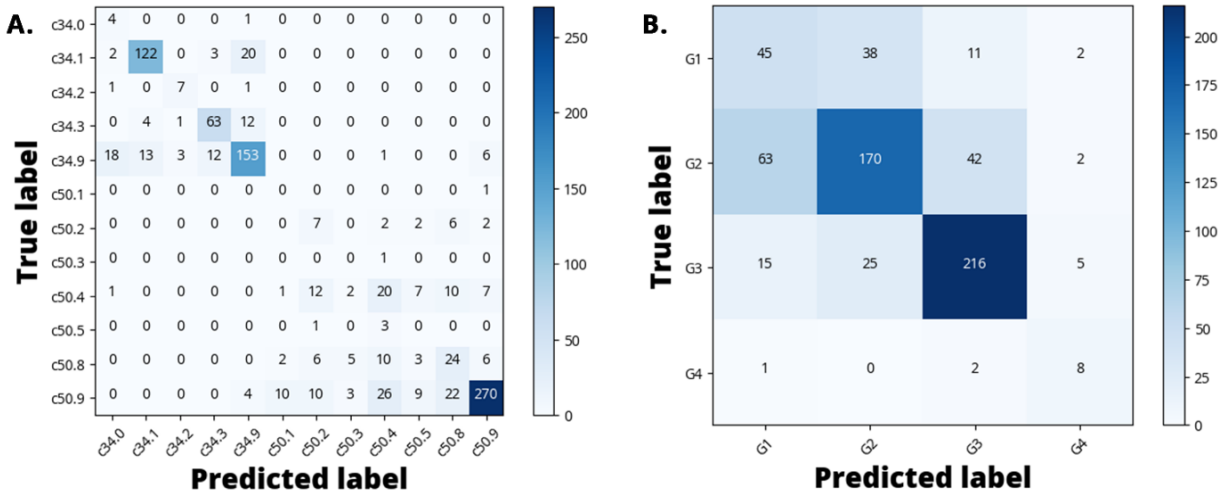


Figure A.10: Extreme Gradient Boosting Confusion Matrix for (A.) Primary Site and (B.) Histological Grade

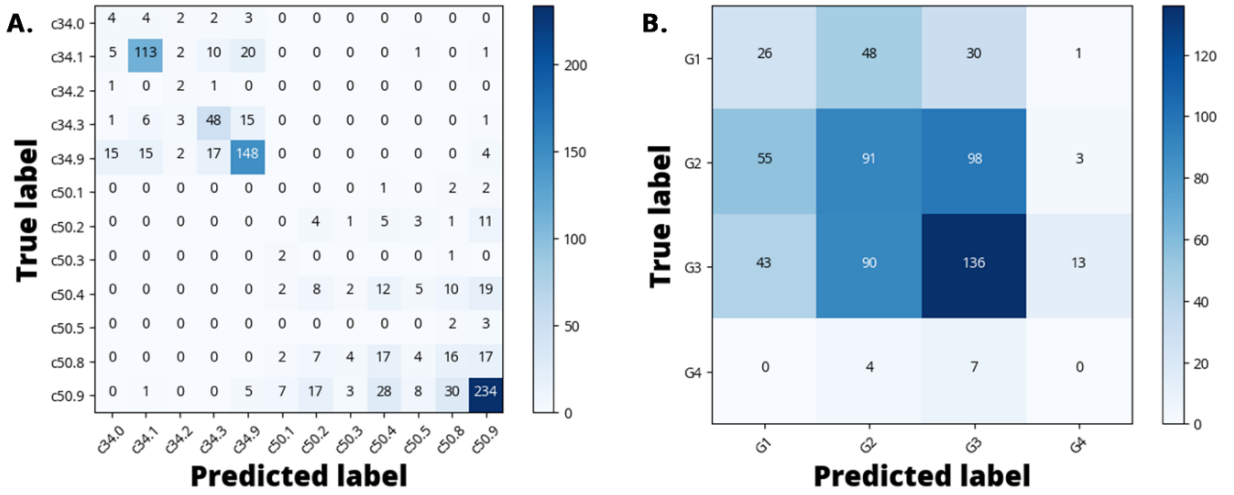


Figure A.11: Recurrent Neural Network without Attention Confusion Matrix for (A.) Primary Site and (B.) Histological Grade

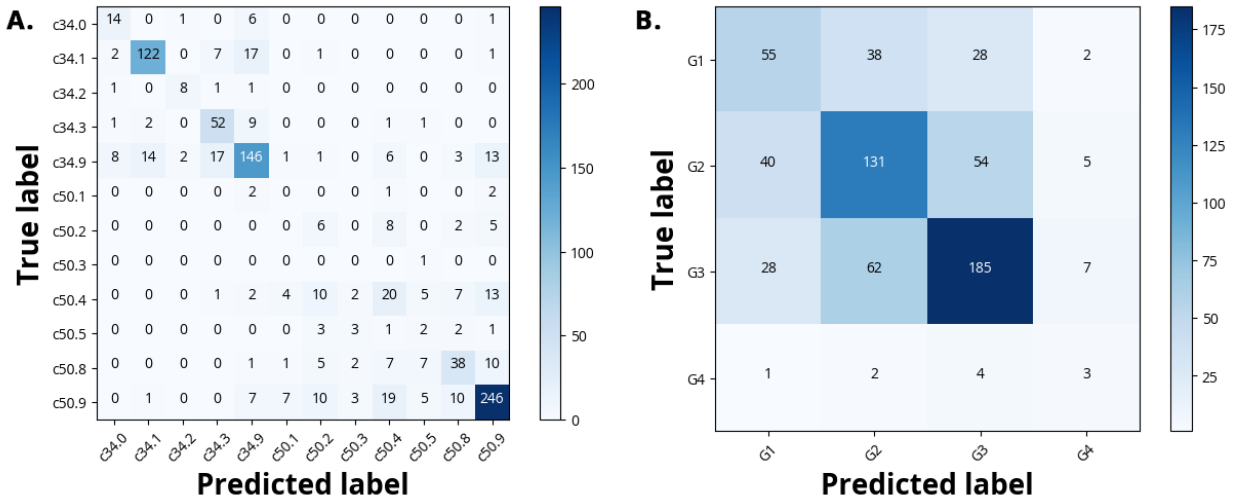


Figure A.12: Recurrent Neural Network with Attention Confusion Matrix for (A.) Primary Site and (B.) Histological Grade

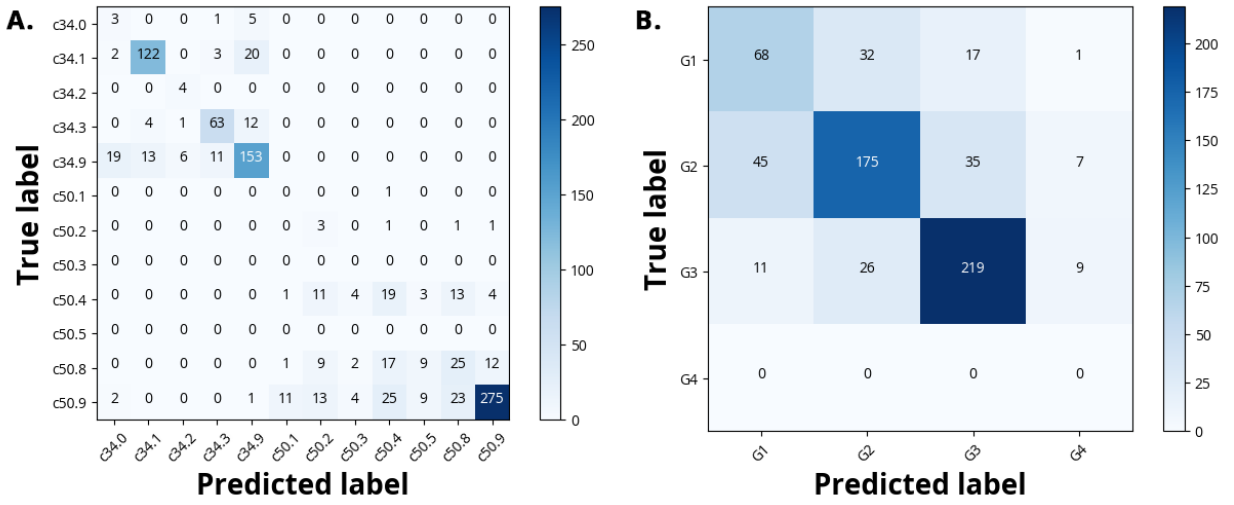


Figure A.13: Convolutional Neural Network Confusion Matrix for (A.) Primary Site and (B.) Histological Grade

A.14 Pubmed Abstract Classification Experiment

One of the limitations of our study was the small size of our dataset. To confirm that the performance of the HAN would improve given more data, we tested the HAN on a simple document classification task with Pubmed abstracts. Each abstract in the Pubmed dataset comes pretagged with medical subject headings (mesh headings) that identify general topics associated with that abstract (e.g. child development, cardiovascular disease). For our task, we only use mesh heading labels that are explicitly listed as major topics for an abstract. We extract Pubmed abstracts associated with 8 selected labelsmetabolism, physiology, genetics, chemistry, pathology, surgery, psychology, and diagnosisand only use abstracts that are associated with a single label. We create three datasets of different sizes, the first with 1,000 documents of each label (8,000 abstracts), the second with 10,000 documents of each label (80,000 abstracts), and the third with 100,000 documents of each label (800,000 abstracts).

We train the same HAN architecture on all three datasets for the sake of speed, we use a simplified architecture with 100-dimension word embeddings, 50 GRU units, and 100 hidden neurons in the attention mechanism. For each of the three datasets, we perform 90/10 train/validation splitting, train the HAN on the training set, and save the HAN parameters after the epoch with the highest validation accuracy. Finally, we report the performance

Table A.4: HAN performance on Pubmed abstract classification based on dataset size

Dataset Size	Classification Accuracy	Convergence Time/Epochs
8,000	70.63 3.10	2514 seconds, 8 epochs
80,000	76.50 2.92	18186 seconds, 4 epochs
800,000	76.88 2.96	269769 seconds, 4 epochs

of the three HAN versions on a hold-out test set of 800 documents below in Table S4. A 95% confidence interval for classification accuracy is calculated using the same bootstrapping method described above (page SI-5), but using standard deviation multiplied by 1.96 instead of using the 2.5 and 97.5 percentile entry. For timing, all models were tested using on a single Intel Core i7-3770 CPU @ 3.40GHz.

We see that increasing the training dataset size also improves the performance of HAN. The gain in classification accuracy between the HAN trained on 8,000 abstracts and 80,000 abstracts is about 6%. Meanwhile, the gain between the HAN trained on 80,000 abstracts and 80,0000 abstracts is less than 1% while this difference is very small, a paired t-test between the two accuracies reveals the difference is still statistically significant with a P-value of ¡ 0.001. We attribute the similar performance of the three HANs to the fact that the distributions of the vocabulary (i.e., unique words used) between the smaller and larger training datasets is nearly identical (Figure A.14). Therefore, even with a substantially smaller training corpus, such as the pathology report dataset used for our main experiment, we believe that the HAN is able to capture the intrinsic features necessary for classification. Our results show that when selecting the amount of data on which to train the HAN, one must balance between the training time and the increase in expected performance we see that while the increase in training time relative to data size is close to linear, the increase in performance diminishes as data size increases.

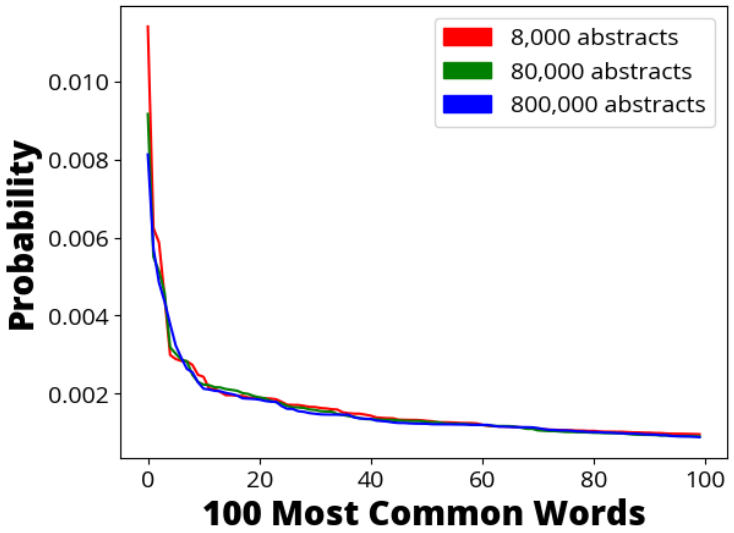


Figure A.14: The probability distributions of the 100 most common words in the Pubmed corpus with 8,000 abstracts, 80,000 abstracts, and 800,000 abstracts

Pubmed 8,000 abstract corpus: 100 most common words: floattoken, largeint, study, patients, using, cell, cells, results, genes, associated, protein, high, studies, used, activity, disease, different, control, clinical, expression, based, data, analysis, showed, cancer, gene, time, related, treatment, species, present, effects, increased, compared, role, risk, model, non, higher, significant, group, important, mitochondrial, tumor, age, factors, region, significantly, genome, human, specific, potential, effect, use, years, findings, health, reported, development, levels, low, changes, children, observed, brain, including, type, cases, complete, identified, coding, diagnosis, population, growth, bp, case, function, patient, report, functional, self, early, response, suggest, research, revealed, new, structure, aim, performed, investigated, long, 10, factor, tissue, life, level, molecular, total, groups

Pubmed 80,000 abstract corpus: 100 most common words: floattoken, largeint, study, patients, cells, cell, using, results, associated, expression, cancer, high, protein, disease, studies, used, analysis, different, clinical, based, data, treatment, gene, showed, activity, role, time, present, related, increased, genes, risk, model, compared, important, human, specific, significant, effects, tumor, group, development, control, factors, significantly, species, including, non, use, potential, levels, new, higher, type, identified, changes, effect, findings,

observed, diagnosis, health, age, low, reported, function, cases, years, patient, molecular, proteins, response, genetic, revealed, 10, brain, case, dna, research, performed, functional, conditions, growth, suggest, level, report, early, investigated, evidence, known, complex, tissue, common, induced, binding, positive, review, factor, structure, number, children

Pubmed 800,000 abstract corpus: 100 most common words: floattoken, largeint, patients, study, cells, cell, results, using, protein, expression, associated, gene, disease, high, used, analysis, different, treatment, clinical, activity, studies, data, showed, cancer, present, based, role, time, human, group, increased, type, related, model, specific, cases, genes, compared, important, significant, tumor, observed, diagnosis, proteins, years, significantly, development, factors, dna, new, patient, use, effects, levels, 10, changes, age, case, control, function, risk, species, binding, reported, low, findings, effect, higher, non, identified, including, report, revealed, suggest, potential, growth, molecular, acid, structure, health, genetic, complex, response, tissue, rate, performed, number, induced, year, early, factor, level, groups, positive, presence, primary, conditions, functional, children, evidence

B Chapter 3 Supporting Information

This appendix contains the supporting information for the paper "Classifying Cancer Pathology Reports with Hierarchical Self-Attention Networks."

B.1 Dataset Distribution and Label Descriptions

Table B.1: Descriptions of the labels used in each classification task. Histology is omitted because there are over 500 labels; these are available in the SEER coding manual at https://seer.cancer.gov/tools/codingmanuals/

Site	Meaning
C00	Lip
C01	Base of tongue
C02	Other and unspecified parts of tongue
C03	Gum
C04	Floor of mouth
C05	Palate
C06	Other and unspecified parts of mouth
C07	Parotid gland
C08	Other and unspecified major salivary glands
C09	Tonsil
C10	Oropharynx
C11	Nasopharynx
C12	Pyriform sinux
C13	Hypopharynx
C14	Other and ill-defined sites in lip, oral cavity, and pharynx
C15	Esophagus
C16	Stomach
C17	Small intestine
C18	Colon
C19	Rectosigmoid junction
C20	Rectum
C21	Anus and anal canal
C22	Liver and intrahepatic bile ducts
C23	Gallbladder
C24	Other and unspecified major salivary glands
C25	Pancreas
C26	Other and ill-defined digestive organs
C30	Nasal cavity and middle ear
C31	Accessory sinuses
C32	Larynx
C33	Trachea
C34	Bronchus and lung
C37	Thymus
C38	Heart, mediastinum, and pleura
C40	Bones, joints and articular cartilage of limbs
C41	Bones, joints and articular cartilage of other and unspecified sites
C42	Hematopoietic and reticuloendothelial systems
C44	Skin
C47	Peripheral nerves and autonomic nervous system
C48	Retroperitoneum and peritoneum
C49	Connective, subcutaneous and other soft tissues
C50	Breast
C51	Vulva
C52	Vagina
C53	Cervix uteri
C54	Corpus uteri
C55	Uterus, NOS
C56	Ovary
C57	Other and unspecified female genital organs
C60	Penis
C61	Prostate gland
C62	Testis
C63	Other and unspecified male genital organs
C64	Kidney
C65	Renal pelvis
C66	Ureter
C67	Bladder
C68	Other and unspecified urinary organs
C69	Eye and adnexa
C70	Meninges
C71	Brain
C72	Spinal cord, cranial nerves, and other parts of central nervous system
C73	Thyroid gland
C74	Adrenal gland
C75	Other endocrine glands and related structures
C76	Other and ill-defined sites
C77	Lymph nodes
C80	Unknown primary site

Laterality	Meaning
0	Not a paired site
1	Right: origin of primary
2	Left: origin of primary
3	Only one side involved, right or left origin unspecified
4	Bilateral involvement at time of diagnosis, lateral origin unknown for a single primary; or both ovaries involved simultaneously, single histology; bilateral retinoblastomas; bilateral Wilms tumors
5	Paired site: midline tumor
9	Paired site, but no information concerning laterality

Behavior	Meaning
0	Benign
1	Borderline malignancy
2	In situ
3	Malignant

Grade	Meaning
1	Grade I; also called well-differentiated
2	Grade II; also called moderately differentiated
3	Grade III; also called poorly differentiated
4	Grade IV; also called undifferentiated or anaplastic
5	T-cell; T-precursor
6	B-Cell; Pre-B; B-precursor
8	NK cell (natural killer cell)
9	Grade unknown, not stated, or not applicable

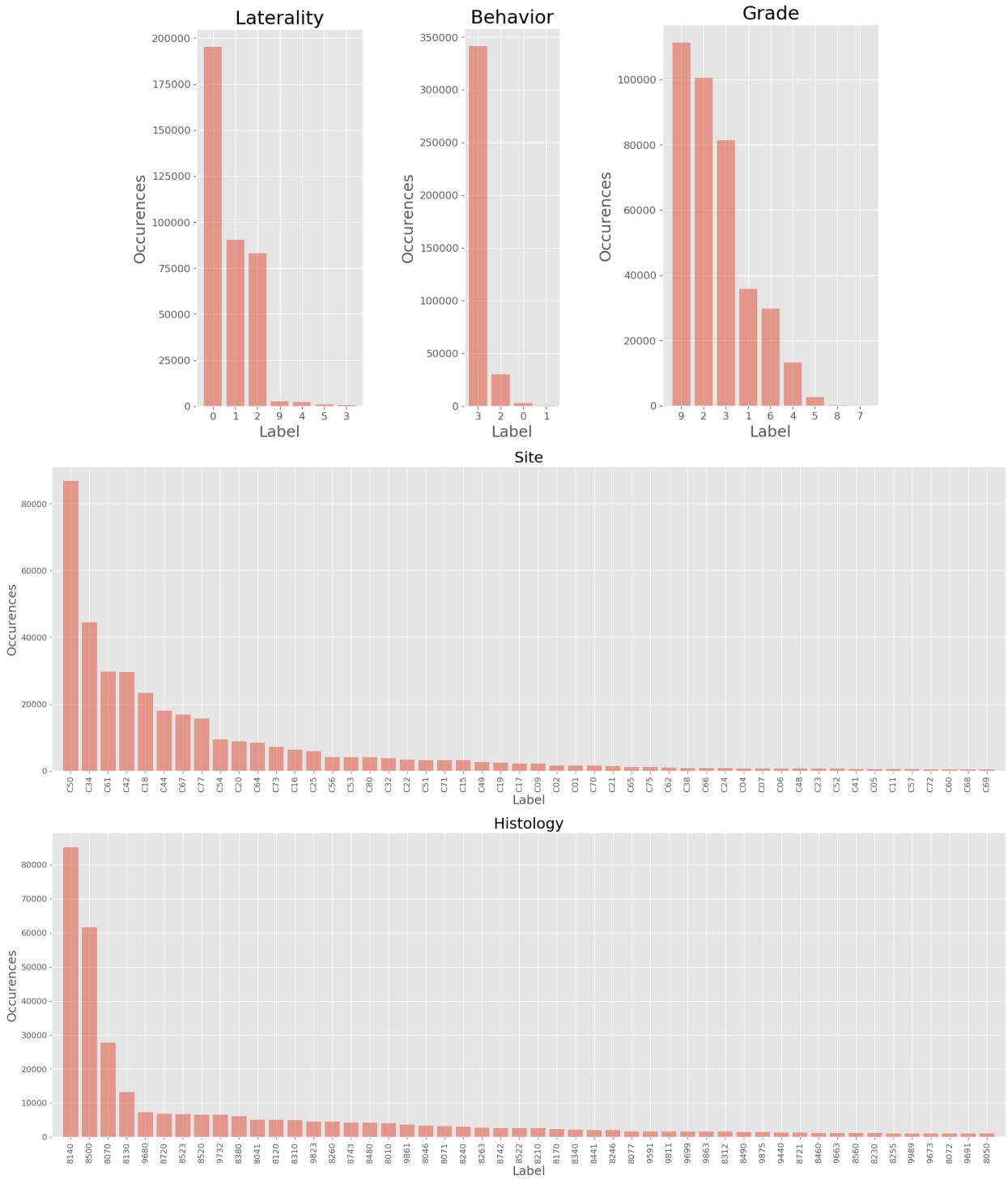


Figure B.1: Histograms of the number of occurrences per label for each of the five classification tasks, arranged from most common to least common. For the site and histology tasks, we only show the 50 most common labels.

B.2 Text Preprocessing and Bootstrapping Details

Pathology Report Preprocessing Procedure

1. Remove identifier segments (registry ID, patient ID, tumor number, and document ID)
2. Remove XML tags
3. Lowercase
4. Replace tabs with spaces, but retain line breaks
5. Standardize oclocks by replacing all references to clock time with a number 1-12 followed by the string "oclock"
6. Remove periods in the abbreviations "dr.", "am.", and "pm." (this allows splitting lines by periods later)
7. Remove periods in floats by replacing all instances of floats with the string "floattoken" (this allows splitting lines by periods later)
8. Replace all integers higher than 100 with the string "largeinttoken" (to reduce the number of unique tokens associated with numbers)
9. Convert unicode to ASCII
10. If the same non-alphanumeric character appears consecutively more than once, replace it with a single copy of that character
11. Add a space before and after every non-alphanumeric character
12. Replace any token that appears less than 5 times across the entire corpus with the string "unknowntoken"
13. Split lines longer than 50 words by any character in the Linebreak Characters Set 1 (listed below)
14. Split lines still longer than 50 words by any character in the Linebreak Characters Set 2 (listed below)

15. Tokenize document, retaining line breaks

16. Replace each token with the appropriate Word2Vec embedding

Linebreak Characters Set 1

- .
- :
- ;
- /
- ?
- ~
- *
- ¡
- #

Linebreak Characters Set 2

- any standalone single letter except 'a' (many reports use single letters to itemize lists)
- ,
- -
- _
- =

Bootstrapping Procedure for Confidence Interval

1. For each model and classification task, save the model's predictions on the test set (hereon referred to as the original predictions)

2. Randomly select predicted labels (with replacement) from the original predictions to create a new set of predicted labels of the same size as the test set (hereon referred to as bootstrapped set)

3. Calculate accuracy and macro F-score on bootstrapped set

4. Repeat steps (2) and (3) 1000 times, saving the scores each time

5. Calculate the 95% confidence interval for accuracy and macro F-score by finding the 2.5 and 97.5 percentile entry for that metric within the 1000 runs (since F-score is not normally distributed)

B.3 Classifier F-Score Breakdown by Class

Table B.2: Classifier f-scores for each task by class. F-scores are calculated using a one-vs-all setup for the class being examined. For histology, we only show the 20 most common classes in the test set. We note that not all possible classes appear in the test sets for each task.

Site	% Total Train Set	% Total Test Set	NB	LR	CNN	HAN	SAN	HiSAN
C00	0.12	0.06	0.00	73.81	74.16	**81.32**	78.16	79.55
C01	0.43	0.43	0.00	56.23	53.51	**57.30**	51.45	55.27
C02	0.46	0.42	0.00	75.51	76.12	**77.98**	73.35	76.50
C03	0.07	0.09	0.00	36.17	26.37	43.48	51.38	**54.55**
C04	0.19	0.21	1.23	63.77	66.44	**69.36**	65.55	64.34
C05	0.15	0.12	0.00	64.20	63.39	**62.00**	58.46	61.11
C06	0.22	0.13	0.00	56.54	52.76	**59.46**	58.15	56.41
C07	0.21	0.18	0.00	64.06	**75.76**	73.36	75.74	75.28
C08	0.05	0.08	0.00	26.19	35.16	32.50	34.04	**36.00**
C09	0.61	0.45	0.00	55.10	59.04	56.34	62.05	**62.58**
C10	0.08	0.15	0.00	16.44	7.69	14.08	17.33	**19.74**
C11	0.15	0.10	0.00	50.42	52.10	**58.02**	49.25	46.05
C12	0.07	0.04	0.00	30.00	39.02	**42.86**	20.51	37.21
C13	0.11	0.07	0.00	**41.86**	22.45	31.46	31.30	40.00
C14	0.06	0.10	0.00	0.00	0.00	**6.90**	0.00	0.00
C15	0.84	0.86	0.30	72.20	72.37	**79.03**	72.33	78.18
C16	1.76	1.46	7.32	75.47	79.07	80.40	75.37	**80.68**
C17	0.58	0.56	0.00	60.70	65.89	71.82	67.97	**77.55**
C18	6.35	5.79	63.65	86.10	88.02	88.92	86.17	**89.54**
C19	0.72	0.65	0.00	33.53	38.16	40.05	36.70	**41.84**
C20	2.39	2.32	0.22	78.68	80.20	81.27	76.74	**81.04**
C21	0.39	0.39	0.00	72.46	76.95	76.27	**78.69**	78.65
C22	0.93	0.82	0.31	77.40	**79.42**	78.81	76.39	79.12
C23	0.17	0.23	0.00	51.11	65.99	65.73	64.41	**70.47**
C24	0.22	0.21	0.00	37.04	38.06	48.11	42.86	**49.84**
C25	1.57	1.56	13.96	83.74	84.15	84.89	83.56	**85.76**
C26	0.09	0.18	0.00	0.00	0.00	2.67	1.24	**8.84**
C30	0.11	0.06	0.00	49.02	52.63	57.41	56.67	**60.19**
C31	0.08	0.08	0.00	35.71	**50.98**	43.96	44.23	48.48
C32	1.08	0.84	16.25	85.97	85.86	85.80	**86.63**	85.78
C33	0.02	0.01	0.00	**18.18**	0.00	0.00	13.33	15.38
C34	12.02	11.40	57.56	88.64	92.14	92.69	92.23	**93.28**
C37	0.04	0.06	0.00	18.75	3.92	44.16	33.33	**53.66**
C38	0.22	0.23	0.00	65.76	71.07	72.22	**74.18**	74.01
C40	0.08	0.06	0.00	46.58	37.67	55.91	63.46	**69.47**
C41	0.17	0.14	0.00	24.83	19.35	**35.58**	35.43	30.86
C42	7.47	9.48	85.69	92.75	94.17	94.40	94.13	**94.69**
C44	4.95	4.50	82.23	94.42	95.23	95.35	95.56	**95.78**
C47	0.04	0.01	0.00	11.11	35.39	**55.17**	21.62	52.17
C48	0.18	0.21	0.00	4.55	10.33	**17.21**	15.45	16.81
C49	0.68	0.71	0.00	64.56	70.34	71.61	71.59	**72.66**
C50	22.91	23.66	84.80	98.30	98.70	98.82	98.75	**98.93**
C51	0.93	0.72	4.80	90.14	91.46	91.12	**92.90**	92.57
C52	0.17	0.15	0.00	61.14	60.96	60.32	**71.22**	69.04
C53	1.21	0.69	3.90	78.93	83.43	83.51	77.83	**86.06**
C54	2.57	2.32	54.37	89.45	90.19	91.31	90.20	**91.99**
C55	0.07	0.07	0.00	0.00	**3.64**	0.00	3.51	0.00
C56	1.10	1.10	0.00	70.86	73.61	73.78	72.15	**74.67**
C57	0.13	0.19	0.00	7.59	2.55	12.27	**12.72**	11.90
C60	0.13	0.12	0.00	85.55	93.48	93.55	**94.12**	93.55
C61	7.83	8.21	95.03	97.60	98.45	98.37	98.17	**98.54**
C62	0.29	0.23	8.51	83.68	88.69	**90.49**	88.57	**90.49**
C63	0.02	0.02	0.00	23.53	0.00	0.00	19.05	**33.33**
C64	2.29	1.98	68.39	92.22	92.93	93.28	92.02	**93.66**
C65	0.31	0.40	0.00	48.99	53.31	51.38	51.30	**55.41**
C66	0.19	0.32	0.00	39.88	42.11	43.58	42.05	**49.88**
C67	4.25	5.86	84.85	93.46	93.79	93.66	93.81	**94.09**
C68	0.09	0.17	0.00	17.57	18.67	15.07	24.69	**25.48**
C69	0.12	0.09	0.00	58.49	58.49	62.61	**72.73**	68.12
C70	0.43	0.34	0.00	93.15	91.67	93.44	92.83	**94.12**
C71	0.90	0.82	3.63	88.09	88.56	90.48	90.74	**91.33**
C72	0.13	0.13	0.00	84.69	80.63	83.17	**87.76**	86.70
C73	2.00	1.78	65.56	95.12	95.73	96.08	95.41	**96.24**
C74	0.08	0.04	0.00	57.14	43.48	**56.25**	48.28	44.44
C75	0.30	0.24	1.04	96.22	92.60	95.49	96.22	**96.77**
C76	0.06	0.04	0.00	0.00	0.00	0.00	**10.81**	8.51
C77	4.23	3.88	53.36	77.65	80.58	81.04	80.20	**82.26**
C80	1.13	0.99	0.00	23.82	25.95	26.09	25.00	**30.77**

Lat.	% Total Train Set	% Total Test Set	NB	LR	CNN	HAN	SAN	HiSAN
0	52.02	52.66	91.11	93.99	94.69	94.78	94.60	**94.89**
1	24.05	23.82	69.73	83.55	83.89	83.85	83.70	**84.79**
2	22.24	22.06	68.94	83.44	83.84	83.56	83.52	**83.81**
3	0.16	0.11	0.00	0.00	0.00	0.00	0.00	0.00
4	0.57	0.53	0.00	19.03	20.42	26.25	**36.63**	33.50
5	0.19	0.25	0.00	26.72	39.41	42.95	44.31	**49.86**
9	0.76	0.58	0.00	2.53	**6.59**	2.43	4.57	3.08

Beh.	% Total Train Set	% Total Test Set	NB	LR	CNN	HAN	SAN	HiSAN
0	0.77	0.61	0.82	89.43	**92.10**	91.74	91.37	91.84
1	0.11	0.10	0.00	59.32	48.92	50.75	60.15	**63.57**
2	8.06	8.58	17.35	76.93	80.12	80.63	79.28	**82.46**
3	91.05	90.70	95.44	97.85	98.10	98.12	98.03	**98.20**

Hist.	% Total Train Set	% Total Test Set	NB	LR	CNN	HAN	SAN	HiSAN
8140	22.70	23.02	57.46	83.54	87.50	88.06	87.58	**88.43**
8500	16.14	17.33	76.72	85.24	87.21	87.34	86.35	**87.83**
8070	7.70	6.26	42.82	81.02	83.00	83.34	81.35	**84.11**
8130	3.28	4.79	45.56	85.88	**85.85**	85.32	84.32	85.31
9732	1.69	2.03	39.28	93.67	94.87	94.44	94.48	**95.01**
9680	1.91	1.93	39.22	72.06	75.07	76.42	77.00	**79.02**
8720	1.85	1.86	46.95	71.20	**73.07**	72.96	71.29	73.04
8520	1.72	1.82	0.00	66.78	70.37	70.56	64.61	**71.53**
8523	1.82	1.68	0.00	31.35	42.73	40.43	41.48	**45.53**
8120	1.32	1.66	0.00	49.80	51.54	**55.57**	52.68	53.79
8380	1.69	1.50	20.44	76.73	79.68	80.73	80.19	**82.02**
9823	1.21	1.49	57.44	82.29	**85.99**	85.71	85.03	85.86
9861	0.85	1.38	11.59	71.82	74.97	76.00	73.23	**76.34**
8310	1.36	1.22	41.73	78.95	80.63	80.81	76.98	**81.78**
8480	1.14	1.21	0.00	45.47	54.48	**58.73**	47.92	57.29
8260	1.23	1.20	4.04	73.55	75.84	75.60	71.82	**77.09**
8041	1.40	1.14	0.00	84.13	86.12	**87.21**	79.93	85.88
8743	1.15	1.10	42.35	69.86	70.93	69.20	**71.01**	70.25
9875	0.24	0.95	1.05	22.98	**44.61**	24.74	30.00	18.64
8240	0.76	0.94	0.27	78.97	79.53	81.61	**82.34**	77.88

Grade	% Total Train Set	% Total Test Set	NB	LR	CNN	HAN	SAN	HiSAN
1	9.20	10.73	0.09	45.39	**61.10**	58.78	49.39	58.93
2	26.58	27.44	55.20	65.76	69.17	70.19	66.20	**70.40**
3	22.40	19.19	24.28	59.36	65.99	65.56	61.04	**66.28**
4	3.37	4.50	0.06	50.17	57.78	57.62	56.98	**58.41**
5	0.76	0.61	0.00	66.24	**87.35**	85.90	79.28	87.11
6	7.65	8.92	74.21	88.76	89.43	90.27	89.05	**90.64**
8	0.38	0.03	0.00	45.71	86.96	**88.89**	78.05	86.96
9	30.04	28.58	64.04	73.45	73.29	75.04	73.26	**75.65**

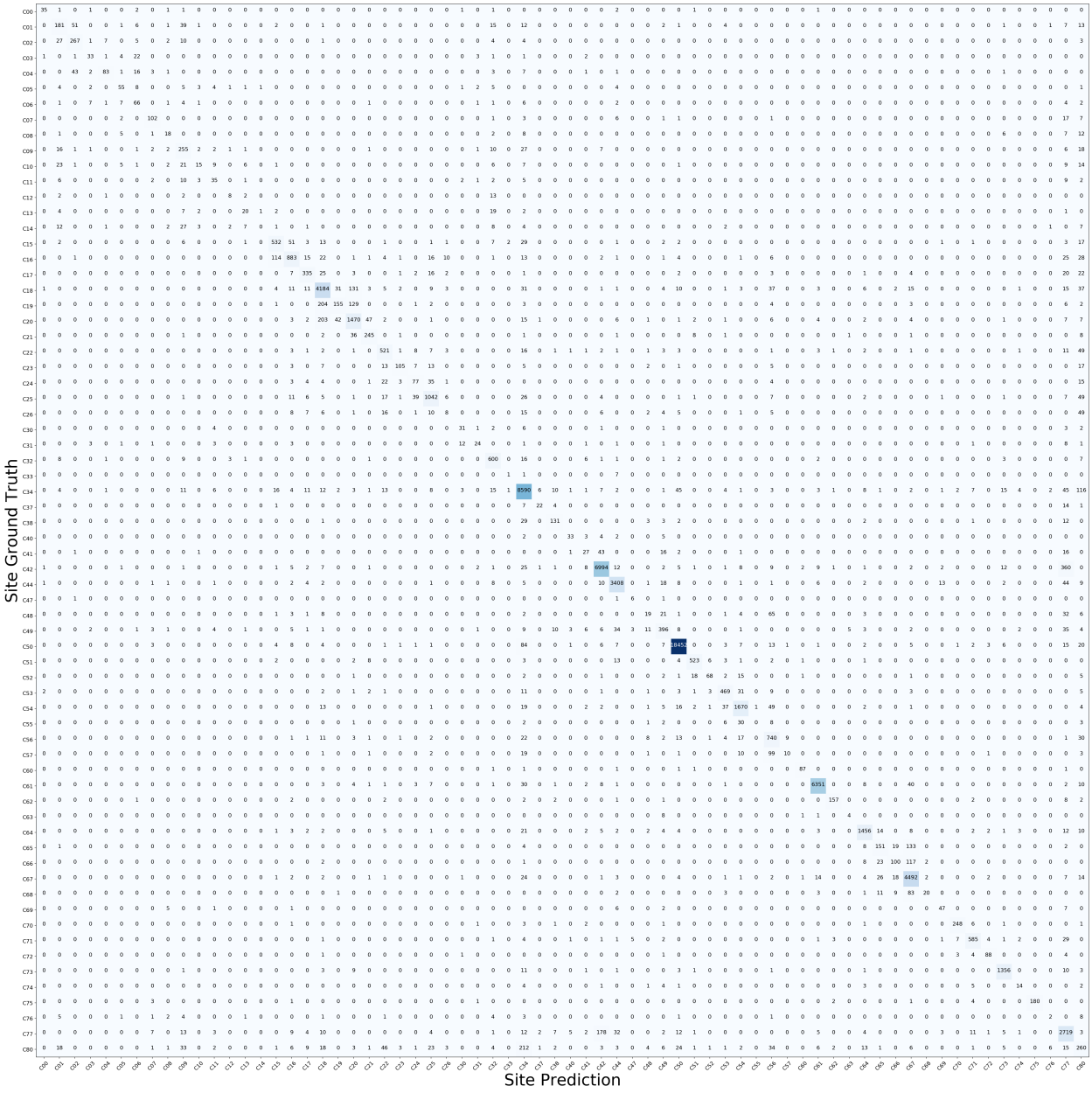


Figure B.2: Confusion matrices for the HiSAN's predictions on the test set for the different classification tasks (continued on following pages). Histology is excluded due to the large number (500+) of possible labels. For each task, we display the confusion matrix with and without normalization by class support size (number of elements in each class).

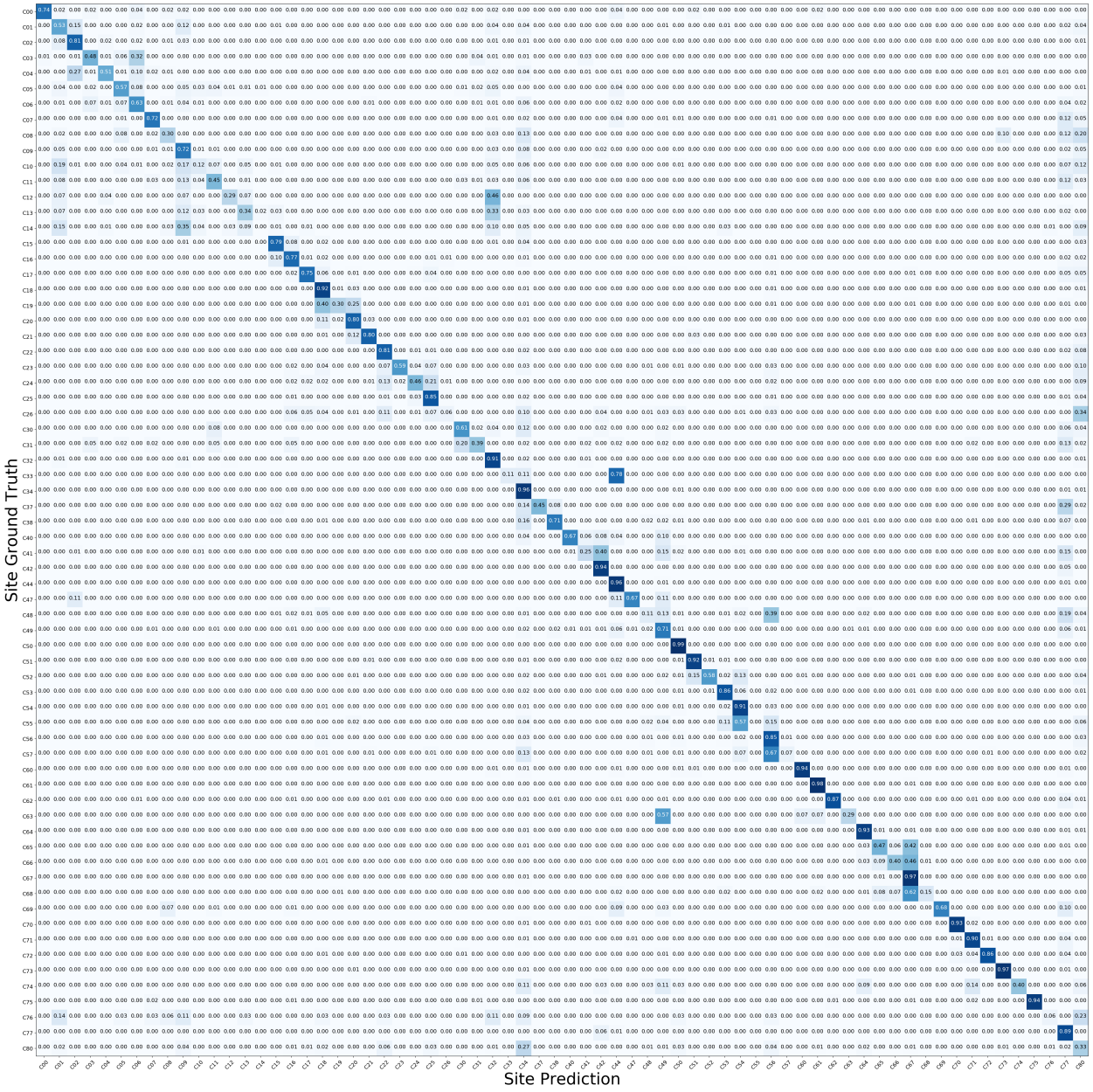


Figure B.2 continued: Confusion matrices for the HiSAN's predictions on the test set for the different classification tasks (continued on following pages). Histology is excluded due to the large number (500+) of possible labels. For each task, we display the confusion matrix with and without normalization by class support size (number of elements in each class).

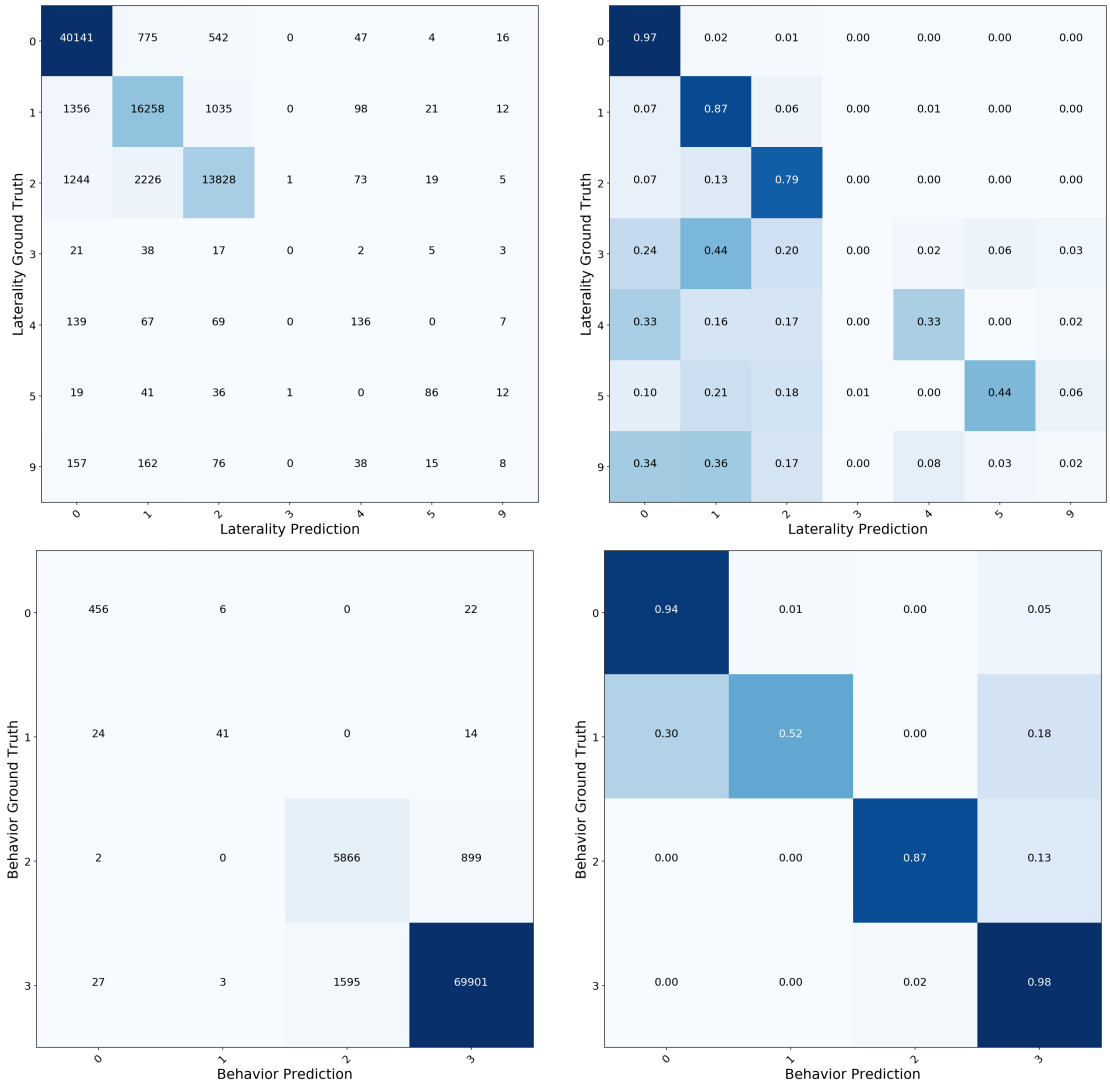


Figure B.2 continued: Confusion matrices for the HiSAN's predictions on the test set for the different classification tasks (continued on following pages). Histology is excluded due to the large number (500+) of possible labels. For each task, we display the confusion matrix with and without normalization by class support size (number of elements in each class).

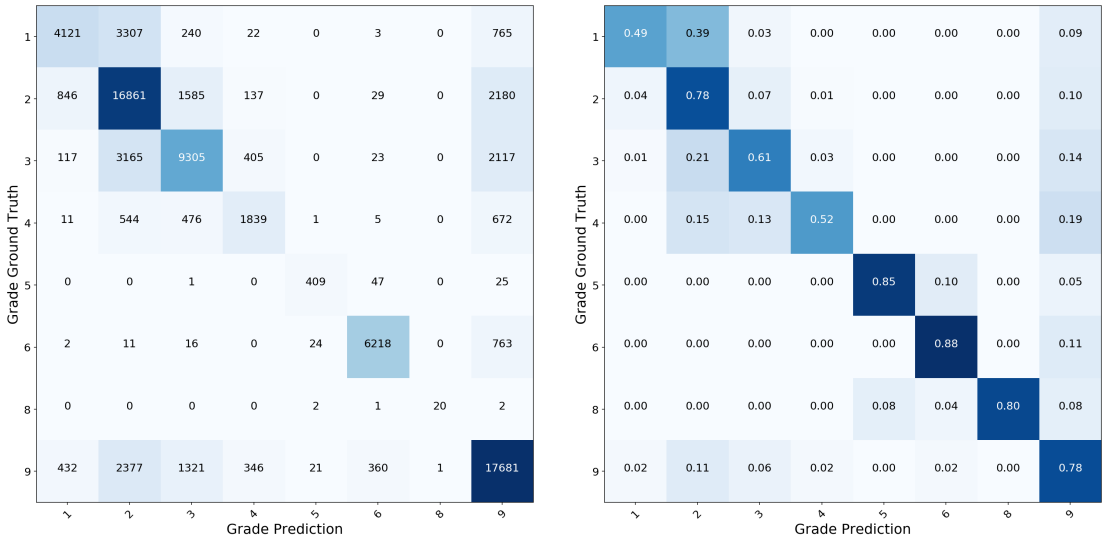


Figure B.2 continued: Confusion matrices for the HiSAN's predictions on the test set for the different classification tasks. Histology is excluded due to the large number (500+) of possible labels. For each task, we display the confusion matrix with and without normalization by class support size (number of elements in each class).

B.4 McNemar's Tests of Significance

We show the statistical significance of the difference in performance of our classifiers using McNemar's Test. For each classification task, we conduct a McNemar's test between the test set predictions (associated with our main set of experiments) of every pair of classifiers. Below, we show the resulting p-values; for any pair of classifiers, we consider the difference to be significant if the p-value is below 0.05. Based off these tests, the HiSAN's performance is significantly different than that of all other classifiers across all classification tasks.

Table B.3: McNemar's test p-values for site task.

	NB	LR	CNN	HAN	SAN	HiSAN
NB	–	0.0000	0.0000	0.0000	0.0000	0.0000
LR	0.0000	–	0.0000	0.0000	0.0000	0.0000
CNN	0.0000	0.0000	–	0.0000	0.0001	0.0000
HAN	0.0000	0.0000	0.0000	–	0.0000	0.0000
SAN	0.0000	0.0000	0.0001	0.0000	–	0.0000
HiSAN	0.0000	0.0000	0.0000	0.0000	0.0000	–

Table B.4: McNemar's test p-values for laterality task.

	NB	LR	CNN	HAN	SAN	HiSAN
NB	–	0.0000	0.0000	0.0000	0.0000	0.0000
LR	0.0000	–	0.0000	0.0000	0.0000	0.0000
CNN	0.0000	0.0000	–	0.7047	0.0095	0.0002
HAN	0.0000	0.0000	0.7047	–	0.0684	0.0000
SAN	0.0000	0.0000	0.0095	0.0684	–	0.0000
HiSAN	0.0000	0.0000	0.0002	0.0000	0.0000	–

Table B.5: McNemar's test p-values for behavior task.

	NB	LR	CNN	HAN	SAN	HiSAN
NB	–	0.0000	0.0000	0.0000	0.0000	0.0000
LR	0.0000	–	0.0000	0.0000	0.0000	0.0000
CNN	0.0000	0.0000	–	0.0608	0.2899	0.0000
HAN	0.0000	0.0000	0.0608	–	0.0041	0.0002
SAN	0.0000	0.0000	0.2899	0.0041	–	0.0000
HiSAN	0.0000	0.0000	0.0000	0.0002	0.0000	–

Table B.6: McNemar's test p-values for histology task.

	NB	LR	CNN	HAN	SAN	HiSAN
NB	–	0.0000	0.0000	0.0000	0.0000	0.0000
LR	0.0000	–	0.0000	0.0000	0.0000	0.0000
CNN	0.0000	0.0000	–	0.0064	0.0000	0.0000
HAN	0.0000	0.0000	0.0064	–	0.0000	0.0000
SAN	0.0000	0.0000	0.0000	0.0000	–	0.0000
HiSAN	0.0000	0.0000	0.0000	0.0000	0.0000	–

Table B.7: McNemar's test p-values for grade task.

	NB	LR	CNN	HAN	SAN	HiSAN
NB	–	0.0000	0.0000	0.0000	0.0000	0.0000
LR	0.0000	–	0.0000	0.0000	0.0000	0.0000
CNN	0.0000	0.0000	–	0.0000	0.0000	0.0000
HAN	0.0000	0.0000	0.0000	–	0.0000	0.0000
SAN	0.0000	0.0000	0.0000	0.0000	–	0.0000
HiSAN	0.0000	0.0000	0.0000	0.0000	0.0000	–

B.5 Classifier Performance on Only Tumor IDs Associated with One Pathology Report

Table B.8: Accuracy and macro F-score (with 95% confidence intervals) on each classification task using a reduced test set of 9481 pathology reports. Reports in this reduced set belong to a Tumor ID that is associated with only a single pathology report, thereby eliminating any addendums and supplementary reports.

	Site		Laterality		Behavior		Histology		Grade	
	Acc	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**
NB	64.7 (63.9, 65.5)	13.6 (13.0, 13.7)	84.1 (83.5, 84.7)	33.3 (32.9, 33.6)	88.7 (88.4, 89.4)	41.4 (40.5, 42.8)	45.8 (45.3, 47.0)	2.43 (2.16, 2.49)	54.8 (54.2, 55.9)	29.8 (29.6, 34.0)
LR	88.3 (88.2, 89.4)	59.8 (59.1, 63.6)	89.5 (89.5, 90.6)	42.9 (40.2, 45.7)	92.2 (91.8, 92.7)	68.0 (60.6, 68.3)	67.1 (66.5, 68.1)	24.0 (20.7, 24.7)	62.2 (61.8, 63.4)	47.4 (46.2, 50.3)
CNN	88.4 (87.8, 88.9)	51.9 (50.4, 54.3)	89.9 (89.4, 90.4)	42.0 (39.4, 44.4)	93.8 (93.4, 94.2)	62.5 (60.8, 64.7)	65.5 (64.7, 66.3)	13.5 (13.2, 16.1)	59.7 (58.8, 60.5)	42.9 (41.0, 46.2)
HAN	89.3 (88.7, 89.7)	59.5 (57.6, 61.7)	90.0 (89.5, 90.5)	41.6 (39.1, 44.1)	94.4 (94.0, 94.8)	70.8 (67.3, 74.0)	72.8 (72.0, 73.6)	23.4 (23.8, 26.9)	63.9 (63.1, 64.8)	60.2 (53.2, 66.2)
SAN	91.4 (90.9, 91.9)	66.9 (64.6, 68.7)	91.5 (91.0, 91.9)	50.7 (47.8, 53.4)	97.5 (97.2, 97.7)	84.8 (81.4, 87.8)	79.4 (78.7, 80.1)	34.4 (34.5, 38.5)	72.9 (72.1, 73.6)	69.8 (62.6, 75.4)
HiSAN split/10 tok	92.1 (91.7, 92.6)	**70.7** (68.1, 72.9)	**92.0** (91.5, 92.4)	**52.2** (49.3, 54.8)	98.0 (97.8, 98.3)	87.9 (84.8, 90.4)	81.3 (80.7, 82.0)	39.3 (39.7, 43.9)	78.9 (78.2, 79.6)	78.5 (68.9, 81.5)
HiSAN split/line	**92.2** (91.7, 92.7)	69.1 (66.9, 71.1)	91.9 (91.5, 92.4)	52.1 (49.2, 54.7)	**98.0** (97.7, 98.2)	**88.8** (85.6, 91.3)	**82.2** (81.6, 82.9)	**39.5** (38.4, 44.0)	**79.5** (78.8, 80.2)	**78.7** (69.0, 81.9)

B.6 Effect of Attention Heads on HiSAN Performance

Table B.9: Test set accuracy and macro F-score (with 95% confidence intervals) on each classification task for experiments on varying the number of attention heads used in the HiSAN.

	Site		Laterality		Behavior		Histology		Grade	
	Acc	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**
HiSAN 1 head	90.1 (89.9, 90.2)	60.3 (59.5, 61.4)	88.8 (88.6, 89.0)	43.4 (42.6, 44.1)	96.4 (96.3, 96.5)	83.8 (81.7, 85.9)	75.7 (75.4, 75.9)	29.2 (29.1, 30.7)	69.8 (69.6, 70.1)	73.1 (71.7, 74.2)
HiSAN 2 heads	90.2 (90.1, 90.4)	61.7 (60.9, 63.0)	89.3 (89.1, 89.4)	50.0 (49.0, 51.0)	96.6 (96.5, 96.7)	84.0 (82.0, 86.1)	75.9 (75.6, 76.1)	29.4 (29.2, 30.9)	71.1 (70.8, 71.3)	74.3 (73.1, 75.3)
HiSAN 4 heads	**90.4** (90.2, 90.5)	63.3 (62.3, 64.0)	89.3 (89.1, 89.5)	50.8 (49.9, 51.7)	96.7 (96.6, 96.8)	83.1 (80.8, 85.3)	76.1 (75.9, 76.4)	**30.7** (30.5, 32.2)	71.4 (71.1, 71.7)	**74.9** (73.7, 75.9)
HiSAN 8 heads	**90.4** (90.2, 90.5)	**63.4** (62.3, 64.1)	89.4 (89.2, 89.5)	50.0 (49.1, 51.0)	**96.7** (96.6, 96.8)	**84.0** (81.6, 86.1)	**76.2** (76.0, 76.5)	30.2 (30.1, 31.8)	**71.6** (71.3, 71.9)	74.3 (73.0, 75.3)
HiSAN 16 heads	90.3 (90.2, 90.5)	63.0 (62.1, 63.8)	**89.5** (89.3, 89.7)	**51.2** (50.3, 52.1)	96.7 (96.6, 96.8)	82.6 (80.3, 84.9)	**76.2** (76.0, 76.5)	30.0 (30.0, 31.6)	71.5 (71.3, 71.8)	74.5 (73.1, 75.6)

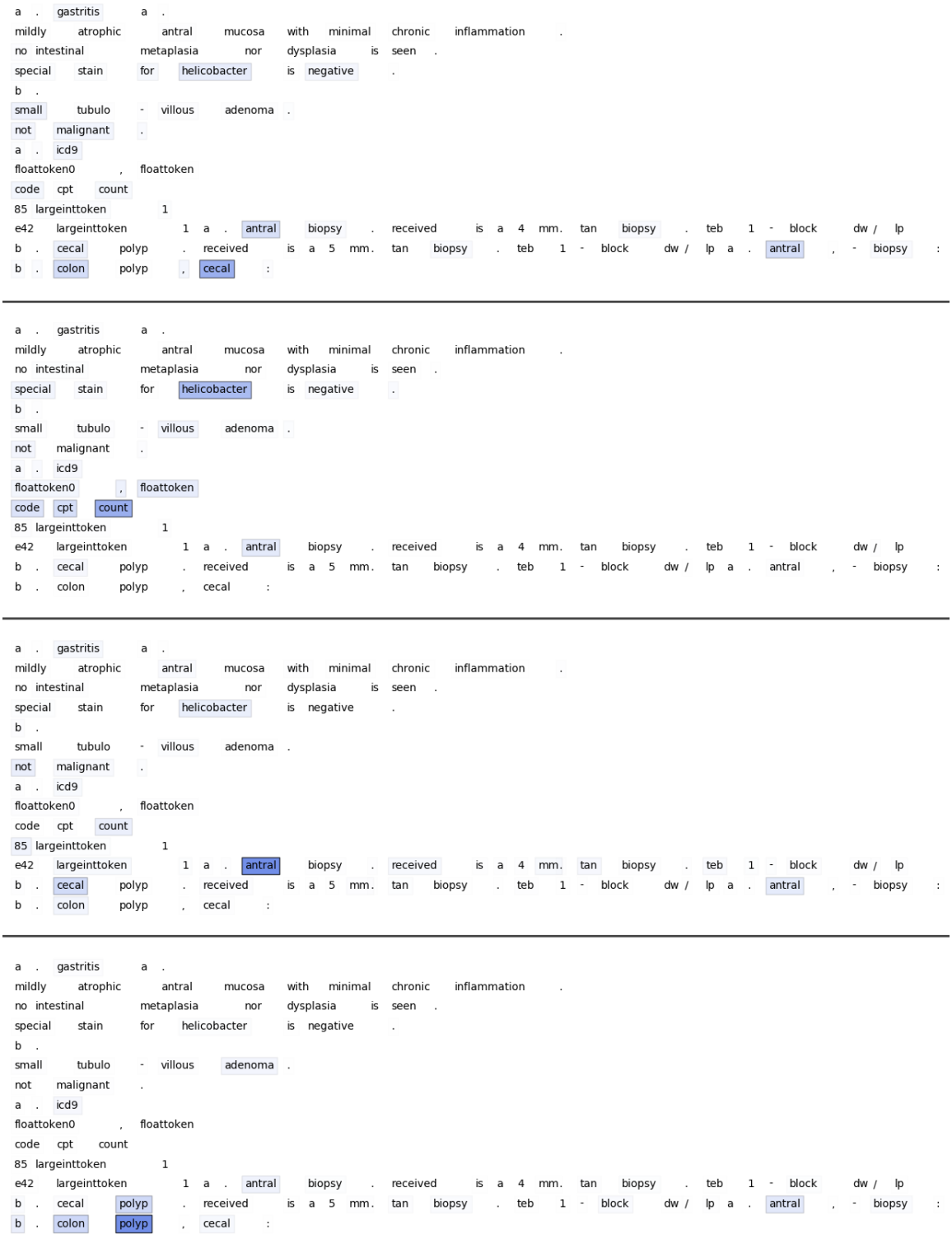


Figure B.3: The attention weights assigned by different attention heads to each word in the same cancer pathology report for the site classification task. We see that different attention heads learn to pay attention to different sets of words.

B.7 Ablation Study of HCAN vs HiSAN

Table B.10: The HiSAN architecture is similar to our previous work on HCANs; however, we found that certain components of the HCAN either decreased performance or provided no consistent improvement on performance when applied to cancer pathology reports – these components include using position embeddings, using a convolution window of size 3 instead of size 1 for Q, K, and V, using convolutions in the target attention, using layer normalization, and using two parallel self-attention layers. The following shows an ablation study of how different HCAN components affect test set accuracy and macro F-score (with 95% confidence intervals) on each classification task.

	Site		Laterality		Behavior		Histology		Grade	
	Acc	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**
HiSAN	**90.4**	**63.4**	89.4	**50.0**	**96.7**	84.0	76.2	30.2	**71.6**	74.3
Original	(90.2, 90.5)	(62.3, 64.1)	(89.2, 89.5)	(49.1, 51.0)	(96.6, 96.8)	(81.6, 86.1)	(76.0, 76.5)	(30.1, 31.8)	(71.3, 71.9)	(73.0, 75.3)
HiSAN	90.3	62.5	**89.4**	48.9	96.7	**85.0**	**76.3**	**30.4**	71.3	**74.9**
Pos Embeds	(90.1, 90.5)	(61.7, 63.3)	(89.3, 89.6)	(47.8, 49.9)	(96.6, 96.8)	(82.9, 86.9)	(76.1, 76.6)	(30.1, 31.8)	(71.0, 71.6)	(73.8, 75.9)
HiSAN	89.8	58.0	89.2	49.1	96.5	77.2	75.3	23.6	70.2	72.2
Conv-3	(89.6, 89.9)	(57.2, 58.7)	(89.0, 89.4)	(48.1, 50.1)	(96.3, 96.6)	(74.7, 79.6)	(75.0, 75.5)	(23.7, 25.0)	(69.9, 70.4)	(70.6, 73.5)
HiSAN	90.1	62.0	89.4	49.4	96.7	82.5	76.0	28.6	71.6	74.9
Target Conv	(89.9, 90.3)	(61.0, 62.7)	(89.2, 89.6)	(48.4, 50.4)	(96.6, 96.8)	(80.1, 84.7)	(75.7, 76.2)	(28.4, 30.1)	(71.3, 71.8)	(73.6, 75.9)
HiSAN	90.2	62.4	89.2	49.3	96.7	83.8	76.2	30.0	70.9	74.0
LayerNorm	(90.1, 90.4)	(61.3, 63.2)	(89.0, 89.4)	(48.2, 50.3)	(96.6, 96.8)	(81.5, 85.8)	(75.9, 76.4)	(29.7, 31.5)	(70.6, 71.2)	(72.7, 75.1)
HiSAN	90.0	60.7	89.4	49.9	96.6	81.7	75.6	28.0	71.5	73.1
Parallel Att	(89.8, 90.2)	(59.9, 61.3)	(89.2, 89.5)	(48.8, 50.9)	(96.5, 96.7)	(79.3, 84.0)	(75.4, 75.9)	(27.8, 29.3)	(71.3, 71.8)	(71.4, 74.4)
HCAN	89.4	57.4	89.2	49.6	96.5	77.4	74.8	22.7	69.7	71.5
Original	(89.2, 89.6)	(56.5, 58.1)	(89.0, 89.4)	(48.5, 50.1)	(96.4, 96.6)	(74.8, 79.9)	(74.6, 75.1)	(22.7, 24.0)	(69.5, 70.0)	(70.0, 72.8)

B.8 Experiments on Vocabulary Size

Table B.11: In our main set of experiments, we replace rare words (appearing fewer than five times in the corpus) with an "unknown_word" token. Below, we also show the performance of the HiSAN in which all possible words are included in the word embedding vocabulary matrix. Including all words gives small boost in performance in the histology task, which has the highest number of possible classes and highest class imbalance; however, it decreases performance on all other tasks.

	Site		Laterality		Behavior		Histology		Grade	
	Acc	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**	**Acc**	**Macro**
HiSAN	**90.4**	**63.4**	**89.4**	**50.0**	**96.7**	**84.0**	76.2	30.2	**71.6**	**74.3**
Min Count 5	(90.2, 90.5)	(62.3, 64.1)	(89.2, 89.5)	(49.1, 51.0)	(96.6, 96.8)	(81.6, 86.1)	(76.0, 76.5)	(30.1, 31.8)	(71.3, 71.9)	(73.0, 75.3)
HiSAN	90.3	62.3	89.3	49.8	96.6	84.2	**76.3**	**31.71**	71.6	73.8
All Vocab	(90.1, 90.5)	(61.5, 63.3)	(89.1, 89.5)	(48.8, 50.7)	(96.5, 96.7)	(82.0, 86.3)	(76.0, 76.5)	(31.47, 33.3)	(71.3, 71.8)	(72.4, 74.9)

C Chapter 4 Supporting Information

This appendix contains the supporting information for the paper "Using Case-Level Context to Classify Cancer Pathology Reports."

C.1 Dataset Distributions

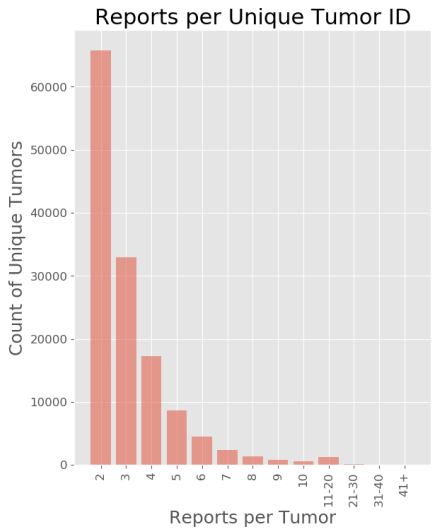


Figure C.1: Histogram of number of pathology reports associated with each unique tumor ID.

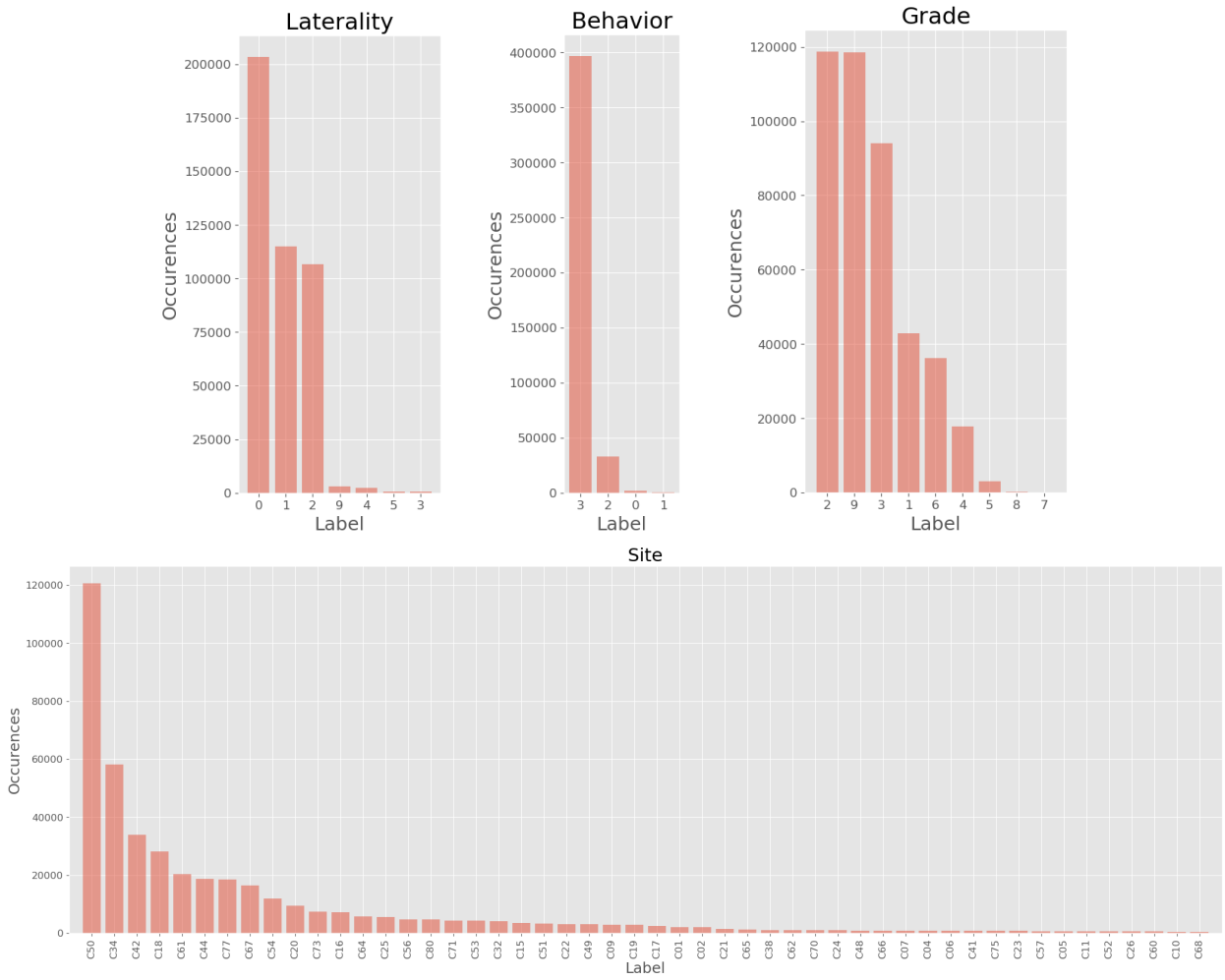


Figure C.2: Histograms of the number of occurrences per label for each of the six classification tasks, arranged from most common to least common. For the site, subsite, and histology tasks, we only show the 50 most common labels. Detailed information about each label can be found online in the SEER coding manual at https://seer.cancer.gov/tools/codingmanuals/. (Figure continued on next page.)

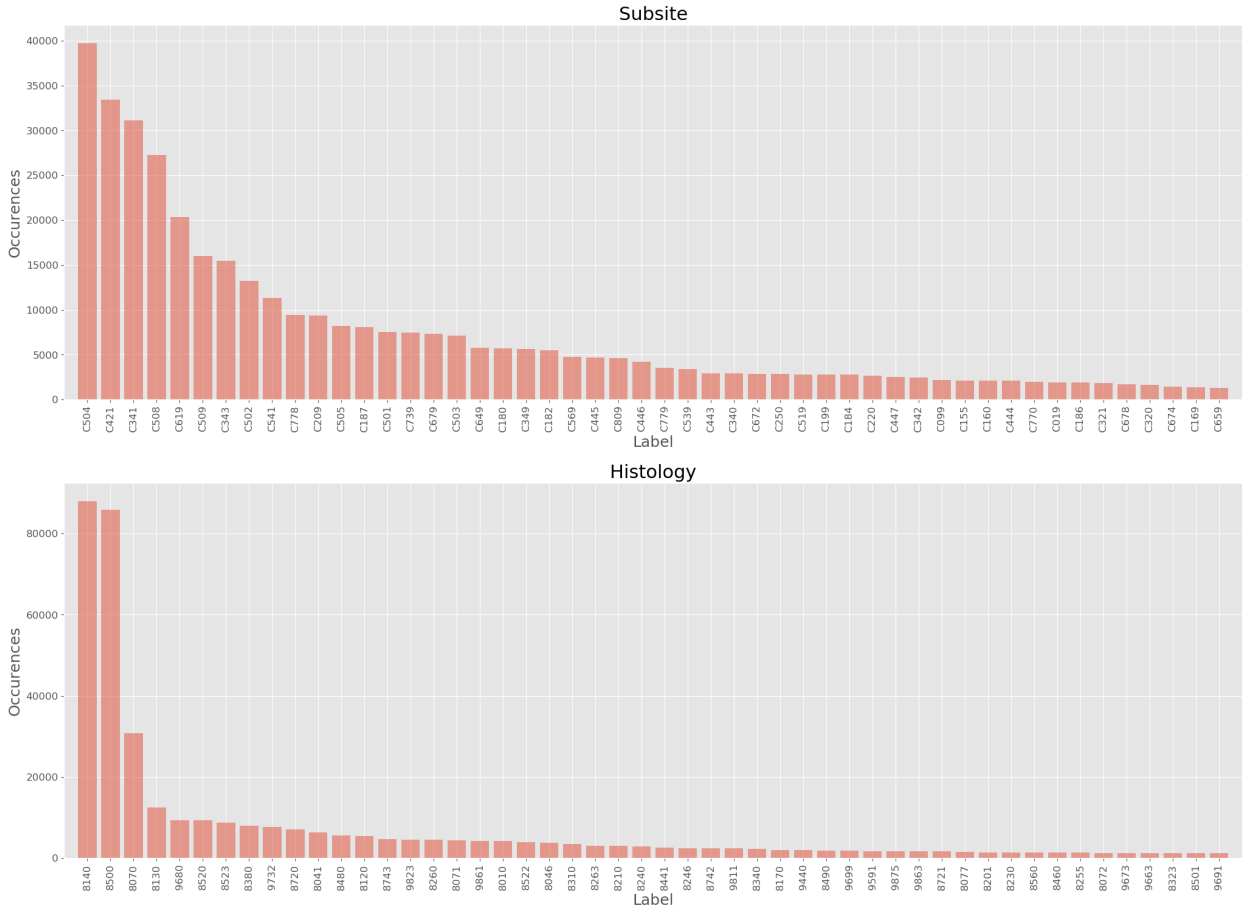


Figure C.2 continued: Histograms of the number of occurrences per label for each of the six classification tasks, arranged from most common to least common. For the site, subsite, and histology tasks, we only show the 50 most common labels. Detailed information about each label can be found online in the SEER coding manual at https://seer.cancer.gov/tools/codingmanuals/.

C.2 Detailed Experimental Procedures

Pathology Report Preprocessing Procedure

1. Remove identifier segments (registry ID, patient ID, tumor number, and document ID)
2. Remove XML tags
3. Lowercase
4. Replace tabs with spaces, but retain line breaks
5. Remove periods in the abbreviations "dr.", "am.", and "pm." (this allows splitting lines by periods later)
6. Remove periods in floats by replacing all instances of floats with the string "floattoken" (this allows splitting lines by periods later)
7. Replace all integers higher than 100 with the string "largeinttoken" (to reduce the number of unique tokens associated with numbers)
8. Convert unicode to ASCII
9. If the same non-alphanumeric character appears consecutively more than once, replace it with a single copy of that character
10. Add a space before and after every non-alphanumeric character
11. Replace any token that appears less than 5 times across the entire corpus with the string "unknowntoken"
12. For the HiSAN input, split the document by naturally occurring linebreaks.
13. For the HiSAN input, split lines longer than 50 words by any character in the Linebreak Characters Set 1 (listed below)
14. For the HiSAN input, split lines still longer than 50 words by any character in the Linebreak Characters Set 2 (listed below)

15. Replace each word token with the appropriate Word2Vec embedding

Linebreak Characters Set 1

- .
- :
- ;
- /
- ?
- ~
- *
- <
- #

Linebreak Characters Set 2

- any standalone single letter except 'a' (many reports use single letters to itemize lists)
- ,
- -
- _
- =

Bootstrapping Procedure for Confidence Interval

1. For each model and classification task, save the model's predictions on the test set (hereon referred to as the original predictions)
2. Randomly select predicted labels (with replacement) from the original predictions to create a new set of predicted labels of the same size as the test set (hereon referred to as bootstrapped set)

3. Calculate accuracy and macro F-score on bootstrapped set

4. Repeat steps (2) and (3) 1000 times, saving the scores each time

5. Calculate the 95% confidence interval for accuracy and macro F-score by finding the 2.5 and 97.5 percentile entry for that metric within the 1000 runs (since F-score is not normally distributed)

C.3 Modular vs End-to-End Training

Table C.1: Accuracy and Macro F-Score (with 95% confidence intervals) of modular and end-to-end training using the CNN baseline model. Results are grouped in pairs – for each technique for capturing report level context, we compare the modular results with the end-to-end results.

	Site	Subsite	Laterality	Histology	Behavior	Grade
CNN w/ BiRNN - Modular	92.37	63.16	**92.28**	**79.59**	98.61	79.72
Accuracy	(91.87, 92.37)	(62.78, 63.68)	**(92.01, 92.51)**	**(78.90, 79.63)**	(98.48, 98.70)	(79.26, 80.00)
	62.14	27.42	49.89	**32.29**	**73.83**	79.03
Macro F-Score	**(60.26, 62.66)**	(26.16, 27.46)	(46.91, 50.33)	**(30.65, 32.56)**	**(69.56, 78.16)**	(78.66, 80.02)
CNN w/ BiRNN - End2End	**92.39**	**63.52**	92.25	78.86	**98.67**	**80.68**
Accuracy	**(91.83, 92.35)**	**(63.12, 64.01)**	(92.17, 92.66)	(78.48, 79.21)	**(98.57, 98.79)**	**(80.67, 81.40)**
	59.21	**27.76**	**50.26**	28.54	72.07	**80.33**
Macro F-Score	(58.96, 91.09)	**(27.40, 28.88)**	**(48.19, 52.68)**	(27.89, 29.66)	(69.63, 78.89)	**(80.11, 81.38)**
CNN w/ Bi-RNN + CRF - Modular	**92.26**	**63.03**	**92.29**	**79.27**	98.64	80.66
Accuracy	**(91.97, 92.45)**	**(62.65, 63.59)**	**(92.21, 92.69)**	**(78.62, 79.35)**	(98.50, 98.71)	(80.53, 81.27)
	64.17	**32.88**	47.22	33.85	76.22	79.31
Macro F-Score	**(63.89, 66.79)**	**(31.87, 33.49)**	(44.75, 51.84)	(33.88, 35.78)	(74.28, 82.98)	(78.79, 80.20)
CNN w/ Bi-RNN + CRF - End2End	91.87	62.45	92.14	78.74	**98.74**	**80.79**
Accuracy	(91.61, 92.11)	(62.18, 63.05)	(91.93, 92.41)	(78.29, 79.03)	**(98.67, 98.88)**	**(80.53, 81.27)**
	60.58	32.87	**49.30**	**34.09**	**80.32**	**80.61**
Macro F-Score	(59.56, 61.79)	(31.99, 33.73)	**(47.69, 52.09)**	**(32.59, 34.21)**	**(78.01, 94.44)**	**(79.95, 81.21)**
CNN w/ Self-Att - Modular	**92.60**	**64.40**	92.49	**80.55**	98.73	**82.68**
Accuracy	**(92.32, 92.79)**	**(63.94, 64.84)**	(92.22, 92.67)	**(79.89, 80.66)**	(98.57, 98.78)	**(82.19, 82.87)**
	61.92	30.20	47.52	**35.27**	71.48	**82.55**
Macro F-Score	(60.75, 62.80)	(29.73, 31.20)	(46.36, 50.64)	**(34.02, 35.73)**	(70.06, 79.63)	**(81.59, 82.70)**
CNN w/ Self-Att - End2End	92.40	64.21	**92.84**	80.08	**98.87**	81.74
Accuracy	(91.71, 92.22)	(64.13, 65.03)	**(92.36, 92.84)**	(79.55, 80.28)	**(98.75, 98.94)**	(81.17, 81.89)
	62.90	**30.77**	**51.35**	33.19	**77.40**	81.32
Macro F-Score	**(60.92, 63.06)**	**(30.37, 31.80)**	**(49.44, 58.75)**	(32.62, 34.37)	**(70.00, 78.23)**	(81.08, 82.13)
CNN w/ Self-Att + CRF - Modular	**92.30**	**62.53**	92.15	**78.81**	98.79	**82.08**
Accuracy	**(92.12, 92.60)**	**(62.15, 63.06)**	(91.95, 92.45)	**(78.33, 79.07)**	(98.72, 98.92)	**(82.18, 82.87)**
	65.41	**34.46**	49.29	**37.62**	77.28	81.27
Macro F-Score	**(64.67, 67.70)**	**(33.00, 34.62)**	(46.16, 53.53)	**(36.09, 37.81)**	(74.97, 84.09)	(80.79, 82.13)
CNN w/ Self-Att + CRF - End2End	92.03	61.38	**91.42**	77.72	**98.82**	81.95
Accuracy	(91.60, 92.10)	(60.89, 61.77)	**(91.06, 91.56)**	(77.14, 77.89)	**(98.71, 99.01)**	(81.27, 82.15)
	63.62	33.99	**51.00**	34.81	**82.10**	**82.07**
Macro F-Score	(62.27, 64.53)	(33.00, 34.66)	**(48.81, 51.51)**	(33.83, 35.56)	**(75.71, 84.27)**	**(80.64, 82.87)**
CNN w/ RNN - Modular	**90.60**	61.88	91.43	**76.01**	97.96	**76.49**
Accuracy	**(90.39, 90.91)**	(60.99, 61.92)	(91.20, 91.73)	**(75.55, 76.32)**	(97.81, 98.07)	**(76.15, 76.93)**
	56.78	**26.11**	45.73	**28.79**	71.15	**76.80**
Macro F-Score	**(55.68, 58.01)**	**(24.84, 26.10)**	(44.30, 52.28)	**(28.03, 29.77)**	(69.74, 78.59)	**(75.77, 77.22)**
CNN w/ RNN - End2End	90.33	**62.16**	**91.56**	75.64	**98.15**	76.35
Accuracy	(90.16, 90.69)	**(61.71, 62.61)**	**(91.19, 91.70)**	(75.72, 76.51)	**(97.92, 98.18)**	(76.01, 76.81)
	55.54	25.06	**46.37**	27.83	**73.44**	76.14
Macro F-Score	(54.05, 56.20)	(24.27, 25.48)	**(45.41, 50.89)**	(26.85, 28.55)	**(72.24, 81.84)**	(74.43, 75.95)
CNN w/ RNN + CRF - Modular	**90.82**	**61.50**	91.37	**76.53**	**98.32**	77.23
Accuracy	**(90.56, 91.09)**	**(60.73, 61.63)**	(91.25, 91.78)	**(76.07, 76.85)**	**(98.18, 98.41)**	(76.98, 77,72)
	60.19	30.24	**47.65**	32.57	73.05	76.11
Macro F-Score	**(59.01, 61.86)**	(29.71, 31.37)	**(45.04, 48.61)**	(31.21, 33.01)	(69.10, 78.35)	(75.92, 77.29)
CNN w/ RNN + CRF - End2End	90.60	61.14	**91.56**	76.50	98.10	**78.46**
Accuracy	(90.40, 91.12)	(60.99, 91.90)	**(91.42, 91.91)**	(76.14, 76.93)	(97.93, 98.18)	**(78.08, 78.86)**
	58.25	**32.79**	47.37	**33.53**	**76.56**	**78.46**
Macro F-Score	(57.40, 59.73)	**(31.73, 33.39)**	(46.51, 54.39)	**(31.89, 33.78)**	**(69.24, 78.65)**	**(77.37, 78.69)**
CNN w/ Masked SA - Modular	90.63	**61.72**	**91.35**	76.66	**98.19**	76.88
Accuracy	(90.36, 90.88)	**(60.89, 61.82)**	**(90.90, 91.45)**	(75.92, 76.71)	**(97.91, 98.17)**	(76.33, 77.13)
	59.48	**29.42**	47.44	**30.67**	71.33	76.69
Macro F-Score	(57.80, 60.40)	**(27.78, 30.30)**	(45.02, 49.31)	**(29.53, 31.32)**	(68.46, 77.09)	(75.68, 77.12)
CNN w/ Masked SA - End2End	**90.77**	61.53	91.32	**76.79**	98.16	**77.30**
Accuracy	**(90.19, 90.84)**	(60.97, 61.84)	(91.16, 91.66)	**(75.96, 76.78)**	(97.99, 98.25)	**(76.50, 77.46)**
	60.58	27.39	**49.26**	30.56	**76.67**	**77.27**
Macro F-Score	**(59.27, 61.76)**	(26.76, 28.30)	**(49.13, 57.61)**	(28.84, 30.97)	**(75.44, 82.82)**	**(76.07, 77.40)**
CNN w/ M. Self-Att + CRF - Modular	**91.06**	**62.00**	**91.84**	**77.08**	98.40	**80.54**
Accuracy	**(90.88, 91.41)**	**(61.55, 62.42)**	**(91.38, 91.89)**	**(76.50, 77.27)**	(98.32, 98.54)	**(80.14, 80.88)**
	61.09	30.98	46.11	**33.95**	78.66	**79.92**
Macro F-Score	(60.20, 63.52)	(30.71, 32.37)	(44.62, 53.20)	**(32.86, 34.69)**	(71.72, 80.88)	**(79.09, 80.43)**
CNN w/ M. Self-Att + CRF - End2End	90.80	60.60	91.82	75.98	**98.57**	78.56
Accuracy	(90.39., 90.91)	(59.71, 60.72)	(91.55, 92.06)	(75.12, 76.12)	**(98.40, 98.62)**	(78.08, 78.85)
	61.28	**31.81**	**47.22**	32.61	**81.22**	78.92
Macro F-Score	**(58.53, 61.03)**	**(30.92, 32.48)**	**(45.67, 48.37)**	(31.34, 33.14)	**(75.20, 81.98)**	(78.20, 79.42)

Vita

Shang Gao grew up in Atlanta, Georgia and attended college at Duke University, where he attained a bachelors degree in economics. He worked as a technical writer at a software company before returning to graduate school at the University of Tennessee, Knoxville to attain his doctorate degree in data science and engineering. Shang won the most novel solution award at the 2017 Smokey Mountain Data Challenge, was awarded the Bredesen Center's entrepreneurship award in 2018, is an alumni of the Fall 2018 NSF I-Corps south regional cohort, and has a patent for applying machine learning to predictive dialing.

Shang currently works in the Biomedical Sciences, Engineering, and Computing group at Oak Ridge National Laboratory in Oak Ridge, Tennessee, where his research focuses on deep learning for biomedical natural language processing. His work on hierarchical attention networks was selected as one of Oak Ridge Leadership Computing Facilitys (OLCF) science highlights. He is interested in all things deep learning and has worked on deep learning applications in computer vision, autonomous driving, and bioinformatics. In his free time, Shang is an avid reader and gamer.