BDH Final Project: Learning to Diagnose using Clinical Notes

Jinmiao Huang[1]**, Cesar Osorio**[1]**, Luke Wicent Sy**[1]
[1] **Georgia Institute of Technology, Atlanta, Georgia, USA**

Abstract

Clinical notes contain detailed interpretation of patients clinical information. The semantic analysis of such text is important and valuable in improving medical care systems. In this project, we apply deep learning based natural language processing frameworks to automatically assign clinical ICD-9 codes from those free-text clinical notes. Our best models are able to predict the top 10 ICD9 codes with 69.57% F1 and 89.67% accuracy (GRU); the top 10 ICD9 categories with 72.33% F1 and 85.88% accuracy (GRU); the top 50 ICD9 codes with 36.62% F1 and 91.48% accuracy (LR); the top 50 ICD9 categories with 43.01% F1 and 88.41% accuracy. A video summarize of our work can be found at ____. We made our evaluation tools and resources available at https://github.com/lsy3/clinical-notes-diagnosis-dl-nlp.

1 Introduction

Electronic health record (EHR) data capture variety of patient clinical information such as medical history, vital signs, lab test results, clinical notes, etc. EHR data build a continuous flow of information between the doctor and the patient. Clinicians can perform better diagnosis after a detailed understanding of the patient's medical history, disease progression and other symptoms and descriptions from clinic notes. Systematic reviews have shown the clinical care quality improvement using predictive analysis based on EHR data[1].

Some work use structured biosignal features in EHR to build the clinical decision making systems[2,3]. Others mining unstructured free-text to predict the diagnosis[4,5]. Consider the fact that more than 80% of all health record data is in unstructured text[6], these information contain detailed interpretation of the overall clinical information as well as big challenge in predictive analysis because of the unstructured nature. Thus, our work will mainly focus on exploring useful information from clinical notes and automatically assign diagnose codes (ICD-9) to the patient. Our goal is to develop an auto diagnostic system that learns from available EHR data, finds better disease or treatment progress patterns through statistical analysis, and provides optimal suggestions for clinical decision-making.

Recently, deep learning approaches have shown a significant improvement on many Natural Language Processing (NLP) tasks such as language translation, image caption and sentiment analysis. Deep learning models can often be trained end-to-end without any domain-specific and often tedious hand-designed feature engineering. Therefore, we will develop our diagnose learning system using the state-of-the-art NLP deep learning models.

2 Related Work

Currently, there are two major categories of approaches for automatically assigning ICD-9 codes using text-free clinical notes. One is rule-based systems, the other one is machine learning-based methods. Rule-based systems are designed by human experts. This type of methods have out-performed other methods in many cases[7,8]. However, this kind of system heavily relies on the manual intervention of the medical professionals, thus hard to maintain and scale up to more general cases. Our work focuses on the machine learning-based automatic approach which do not require any domain knowledge from the medical experts. In this section, we will briefly review work in learning based approach.

Chen et al.[4] use semantics analysis method which includes dependency parsing of clinical records and the calculation of semantic matching score. There are also many research work on exploring traditional machine learning methods to clinical notes classification, such as Sequential Minimal Optimization[9], Support Vector Machine[10–13].

More recently, Nigam[14] applied deep learning techniques to classify ICD-9 code using clinical notes. Prakash et al.[5] used Condensed Memory Network with a pre-learned memory representation from a knowledge base (medical articles on Wikipedia) to do the same task.

3 Methodology

Figure 1 shows an overview of our methodology pipeline. Our methodology involves the following steps: data preprocessing, feature extraction, and lastly, model training and testing. Specifically, we ran the experiments using Azure virtual machines (NC24 with K80 GPU). Furthermore, spark was used for data preprocessing; spark, sklearn, and gensim for feature extraction; and spark ML, keras, and tensorflow for model training and testing. Section 3.1 to 3.3 describes each step in more detail. Each model is also evaluated under a set of metrics, as described in section 3.4.

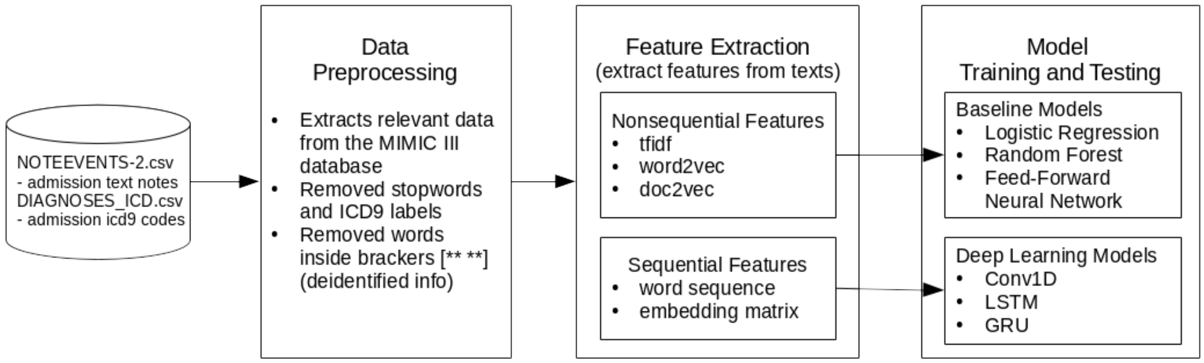


Figure 1: Methodology Pipeline Overview

3.1 Data Preprocessing

MIMIC III dataset is a large data set relating to patients admitted to critical care units at a large tertiary care hospital. It contains de-identified medical records of patients who stayed within the intensive care units at Beth Israel Deaconess Medical Center from 2001 to 2012[15]. The goal of this study is to explore useful semantic information using unstructured data, therefore only the free-text clinic note section from the dataset, specifically the *noteevents* table, was used. Furthermore, we focused on the *discharge summaries* category as it contains actual ground truth and free-text compared to other categories. This approach is similar to Prakash et al[5] . Since *discharge summaries* were written after the diagnosis was made, the notes are sanitized by removing any mention of class-labels (icd9 codes).

Table 1 describes the number of unique patients, hospital admissions, ICD9 codes and ICD9 categories involve in the MIMIC III dataset. *All MIMIC III* describes the whole dataset, while *noteevents* and *discharge summaries* describe the corresponding subsets.

Coverage	Patients	Hospital Admissions	ICD9 Codes	ICD9 Categories
All MIMIC III	46520	58976	6984	943
noteevents	46146	58361	6967	943
discharge summaries	41127	52726	6918	942

Table 1: MIMIC III Descriptive Statistics

The data was preprocessed to produce separate datasets through two approaches. The first approach is to treat the ICD-9 code independently from each other, find the admissions (unique HADM_ID) for each ICD-9 classification, and consider only records related to the top 10 and top 50 common ICD-9 codes. Top 10 and top 50 are chosen because they cover majority of the dataset (76.9% and 93.6% as can be observed in table 3, and for ease of comparison with the result of[14]. The second approach is to group ICD-9 codes into categories based on its hierarchical nature, with categories for larger sets of similar health conditions (e.g, "Cholera due to vibrio cholerae" has the ICD9 code 001.0, and is categorized as a type of Cholera, which in turn is a type of Intestinal Infectious Disease), and then find the patients for top 10 and top 50 common categories. Evaluation will be separately performed on the four datasets. The four datasets will hereby be referred as *top-10-code*, *top-50-code*, *top-10-category* and *top-50-category* respectively.

Table 2 shows the top 10 ICD9 codes and top 10 ICD9 categories. Table 3 also describes the number of unique hospital admissions related to the four datasets mentioned in the earlier paragraph.

ICD9 Code	Admissions
4019: Hypertension	20046
4280: Congestive heart failure	12842
42731: Atrial fibrillation	12589
41401: Coronary atherosclerosis	12178
5849: Acute kidney failure	8906
25000: Diabetes Type II	8783
2724: Hyperlipidemia	8503
51881: Acute respiratory failure	7249
5990: Urinary tract infection	6442
53081: Esophageal reflux	6154

ICD9 Category	Admissions
401: Essential hypertension	20646
427: Cardiac dysrhythmias	16774
276: Disorders of fluid electrolyte	14712
272: Disorders of lipoid metabolism	14212
414: Other chronic ischemic heart disease	14081
250: Diabetes mellitus	13818
428: Heart failure	13330
518: Other diseases of lung	12997
285: Other and unspecified anemias	12404
584: Acute kidney failure	11147

Table 2: Top 10 ICD9

Data Set	Hospital Admissions	*discharge summaries* Coverage (%)
top-10-code	40562	76.93%
top-50-code	49354	93.60%
top-10-category	44419	84.24%
top-50-category	51034	96.79%

Table 3: Dataset Descriptive Statistics

The filtered datasets will be split to 50-25-25 for training, validation and testing.

3.2 Feature extraction

We use two approaches for feature extraction: bag of words and word2vec/doc2vec. We will use bag of words as a baseline, and compare the results to word2vec/doc2vec.

To create the bag of word features, we tokenized all the notes in the filtered training data set, create a word-document matrix with the count of each word in each note and calculate the TF-IDF for each word. We used two TFIDF configurations: (1) one with top 40,000 words with highest TF-IDF scores as the bag of word features and (2) a second one with minimum document frequency of 10 and maximum document frequency of 0.8, which reduces our total number of words to arund 20,000 words.

word2vec[16,17] takes a tokenized text corpus as an input and produces word vectors as output. word2vec is basically a feed forward neural net model , where the non linear hidden layer is removed. We will use as text corpus text notes from MIMIC III. We also use pre-trained word vectors induced from PubMed found on https://github.com/cambridgeltl/BioNLP-2016[18]

3.3 Model Training and Testing

One fundamental assumption adopted by traditional supervised learning algorithms is that each sample has only one label assigned to it. In our problem, each sample has multiple (one or more) ICD-9 codes attached to it. Generally, there are two main categories for tackling the multi-label classification problem[19] (1) problem transformation methods and (2) algorithm adaptation methods. Problem transformation methods transform the multi-label problem into a set of binary classification or regression problems, multiple binary classifiers are trained separately for each label. Algorithm adaptation methods adapt the algorithms to perform multi-label classification in its full form and only one classifier are trained for all the labels.

In this study, we first create three baseline approaches: Linear Regression, Random Forests and Feed Forward Neural Network. Among which, we used problem transformation methods to get the multi-label output for Linear Regression and Random Forest classifiers. Specifically, in order to assign each sample a set of target labels, we simply trained n different models for n different labels, each model independently predicts a mutual exclusive output (0 or 1) for each sample data. For Feed Forward Neural Network, we used algorithm adaptation based methods, since neural network can be easily adapted to multi-label problem by setting up multiple neurons in the network output layer and set each neuron to a target label correspondingly. CNN, LSTM, and GRU also used algorithm adaptation based methods, similar to Feed Forward Neural Network. In this section, we will describe our implemented models in detail.

3.3.1 Baseline Model: Logistic Regression

Our first baseline model is a binomial logistic regression model implemented using Spark ML. For each label (ICD-9 code or category), a separate logistic regression model was trained and each model independently predicts the said label (0 or 1 for the corresponding ICD-9 code or category). We tried different configurations; specifically "no. of iterations" was tuned between 5 to 100. Since only the notes under *discharge summaries* category are used, there is only 1 note per admission. Features are extracted from this note and are used as inputs for this classifier. For *tfidf* and *doc2vec*, the features are directly used as input features. For *word2vec*, the input features are the average of all the feature vectors of the words in the notes.

3.3.2 Baseline Model: Random Forest

Our second baseline model is a random forest model implemented using Spark ML. The same approach and input for the logistic regression were used here (one model for each label). Different configurations were also tried, specifically "tree depth" was tuned between 5 to 30.

3.3.3 Feed-Forward Neural Network

One advantage of Neural Network is that it can be fitted to multi-label problem in just one model with the proper activation function. We implement the feedforward neural network as the baseline for algorithm adaptation based (described in 3.3) multi-label classification problem. We use the same input features and train-test data split as previously described. We use ReLU activation function for all the hidden layers and sigmoid activation function for the output layer. We use binary cross entropy as the loss function and stochastic gradient decent as the optimizer. We tried several neural network models with one to four different hidden layers. For each hidden layer, we tried a combination of neuron size 50, 100, 300, 500 and 1000, a total of seven models. The results for different configurations are detailed in our attached spreadsheet *Experiments.xlsx*. The predicted label for each output is set to 1 for output probability greater than or equal to 0.5 and 0 otherwise. The best performance model is with three hidden layers with size 1000, 500, 100 respectively.

3.3.4 Convolutional Neural Network

Convolutional neural networks (CNN) have achieved remarkable results in image processing related problems. Images have the property of being "stationary", this suggests that the features that we learn at one part of the image can also be applied to other parts of the image[20]. Recently, CNN models have also shown excellent resutls for NLP such as in semantic parsing[21], search query retrieval[22], and sentence classification[23]. Thus we tested on a series of experiments with convolutional neural networks (CNN) for our problem.

The input features for this classifier are N most recent word sequences (taken from the notes). If we don't have enough feature events, we pad zero vectors at the beginning. The word sequence is then converted into vectors using an embedding matrix based on a *word2vec* model. We tried three to ten convolutional layers with size 64, 128, 256 and one to three fully connected dense layers attached to the last convolutional layer with size 4096, 1024 and 128. Among our model architecture setting, the best performed model pipelines are shown in Figure 11 and 12. Under the hardware setting described, the training time for CNN is less than 30 minutes with 500 maximum epochs and early stop if the validation loss doesn't improve for consecutive 10 epochs.

3.3.5 Recurrent Neural Network, LSTM, and GRUs

Finally, we implement Recurrent Neural Networks (RNNs), Long Short Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) using Keras with Tensorflow backend. The input features is the same as we used in CNN. We tried up to three stacked recurrent layers with a combination 64, 128 and 256 units for each layer in our RNNs. We used the well-known LSTM and GRU units for our RNNs. To predict the ICD-9 classification, we only consider the output nodes of the last time step, and apply the same activation function and loss function as we choose for Feed-Forward Neural Network. The best performed mode architecture for LSTM and GRU are shown in Figure 9 and 10. Under the specified hardware setting, the training time for LSTM and GRU is about 6 to 18 hours with 200 maximum epochs and early stop if the validation loss doesn't improve for consecutive 5 epochs.

3.4 Metrics

Combinations of our dataset, feature extraction methods, and models are evaluated under different performance metrics, including precision, accuracy, F-score and recall metrics for multi-label classification. Specifically, the following metrics are used[24]:

$$\text{Precision} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Z_i|} \qquad \text{Recall} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i|}$$

$$F_1 = \frac{1}{n}\sum_{i=1}^{n}\frac{2|Y_i \cap Z_i|}{|Y_i| + |Z_i|} \qquad \text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}$$

where Y_i is the set of predicted labels, Z_i is the set of ground truth labels, and n is the number of samples.

4 Results

This section is organized as follows: baseline results are described in section 4.1, performance under different configurations are described in section 4.2, and our best model performance are described in section 4.3.

4.1 Trivial Model Performance

In addition to the metrics mentioned in 3.4, other metrics such as hamming loss and macro AUC are also often used for multi-label classification problems. However, there is a highly imbalanced relation between non-diagnose and diagnose observations (80/20 or more for most cases). Therefore, the relative high score in some metrics do not indicate the models worked well enough for this task. For example, the performance for a model with all the test samples labeled as non-diagnose (model only predicts zeros or no disease for all the test patients) is presented in table 4. We can see that the accuracy, AUC macro and Hamming loss have non-zero values. Accuracy in particular has high values. Therefore, we focus on improving the performance of the initial zero entries in table 4 (Precision, Recall, and F1) and also recorded accuracy as a casual reference.

Data Set	Precision	Recall	Accuracy	F1	AUC (macro)	Hamming Loss
top-10-code	0	0	0.8025	0	0.5	0.1974
top-50-code	0	0	0.9130	0	0.5	0.0870
top-10-category	0	0	0.7246	0	0.5	0.2753
top-50-category	0	0	0.8734	0	0.5	0.1266

Table 4: Performance with zero initialized data

4.2 Model Performance under Different Configurations

Different model configurations are tried to give us insight into the most appropriate model configuration. Table 5 describes the different methods of feature extraction used. The features extracted are divided into 2 categories: non-sequential and sequential. The non-sequential features include *tfidf*, *word2vec*, and *doc2vec*, which were used in Logistic Regression, Random Forest, and Feed-Forward Neural Network. The sequential features includes *seq* (word sequences) used in conjunction with an embedding matrix based on *word2vec*, which were used in Conv1D, RNN, LSTM, and GRU. Note that *w2v_ave_bio02* and *w2v_ave_bio30* are pre-train word vectors obtained from[18]. *seqv1* is similar to *seqv0* except that stop words are removed from the sequence. For both *seq* configurations, the vector for stop words in the embedding matrix are zeros.

Feature Extraction	Size	Config
tfidfv0	40000	minDocFreq = 3, max_df = 1.0
tfidfv1	20301	minDocFreq = 10, max_df = 0.8
w2v_ave_f100	100	from word2vec model created using MIMIC III text
w2v_ave_f300	300	from word2vec model created using MIMIC III text
w2v_ave_f600	600	from word2vec model created using MIMIC III text
w2v_ave_bio02	200	pre-trained word vectors from[18] (Pubmed win 2)
w2v_ave_bio30	200	pre-trained word vectors from[18] (Pubmed win 30)
doc2vec_f100	100	created using MIMIC III text
doc2vec_f300	300	created using MIMIC III text
doc2vec_f600	600	created using MIMIC III text
seqv0	2000	word sequence + one of the embedding matrix derived from the word2vec.
seqv1	1500	word sequence (w/ stopwords removed) + embedding matrix

Table 5: Feature Extraction Methods

Figure 2 and 3 shows the best model performance of each model-and-feature-extraction-method pair using the *top-10-code* dataset. The raw data are also shown in the appendix (table 14 and 15).

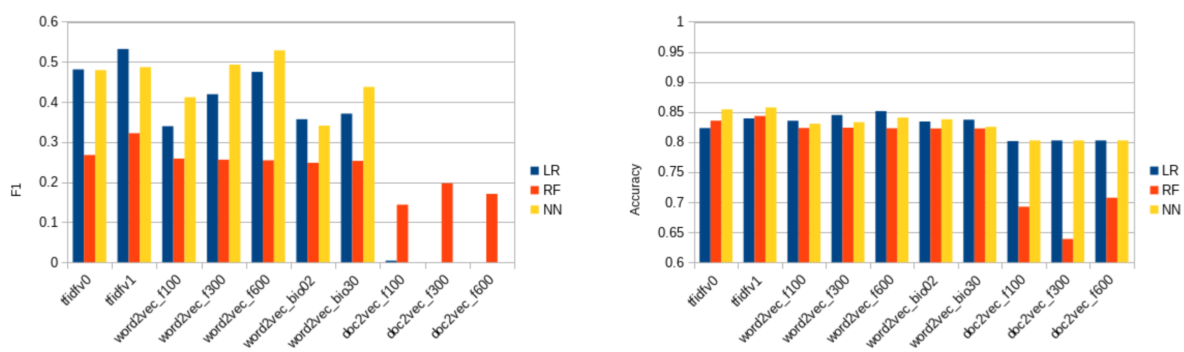


Figure 2: Model Performance under Different Configuration (Non Sequential Data)

For non-sequential feature extraction, *tfidfv1* gave the best f1 results for Logistic Regression and Random Forest, while *w2v_ave_f600* gave the best results for Feed-Forward Neural Networks (although *tfidfv1* also gave a fairly good result for NN). In general, using *tfidfv1* seems to improve performance, maybe due to the reduction of noise and decrease in dimensionality, when compared to *tfidfv0*. *tfidf* configurations generated better results than *word2vec* and *doc2vec*, probably because we are losing information. Note that the feature size for *word2vec* and *doc2vec* are at most 600 while *tfidf* is around 20,000 above. *doc2vec* configurations generated the worse results.

For sequential feature extraction, *seqv0+word2vec_f600* generated the best f1 result for Conv1D, *seqv1+word2vec_bio30* for LSTM, and *seqv0+word2vec_f300* for GRU. In general, all feature extraction methods generated good and comparable results for Conv1D, LSTM, and GRU. It is good to note that our *word2vec* also faired quite well with the

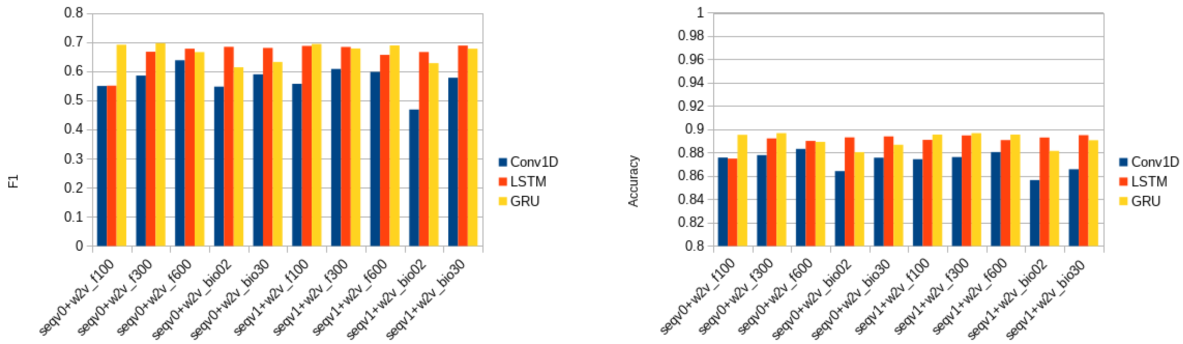


Figure 3: Model Performance under Different Configuration (Sequential Data)

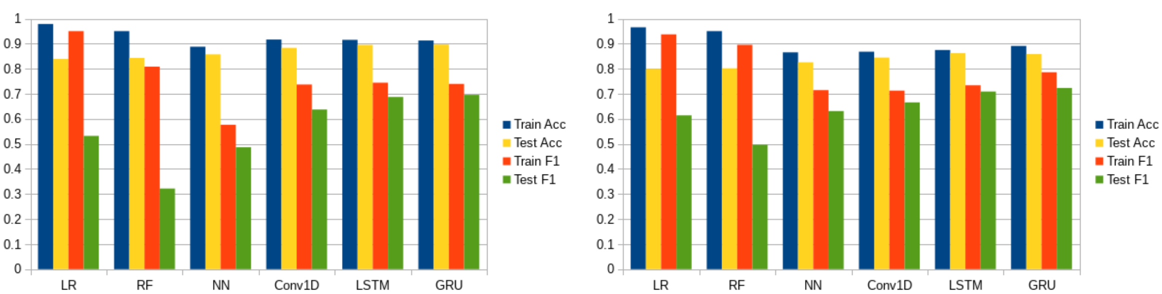
pre-trained *word2vec*. Although not shown in figure 3 (but shown in table 15, the feature extraction methods were also tried for RNN, but the results were bad (0.0 - 0.23 f1 at best). This may be because the sequence length is too long (1,500 - 2,000).

The configuration that provided the best performance for each model is used on the *top-50-code*, *top-10-cat*, and *top-50-cat* datasets. The results are further explained in the next section 4.3.

4.3 Best Model Performance

4.3.1 Overview

Figure 4 and 5 shows the model performance for the *top-10-code*, *top-10-cat*, *top-50-code*, *top-50-cat* dataset. Raw data are also shown in table 8, 11, 9, and 12.



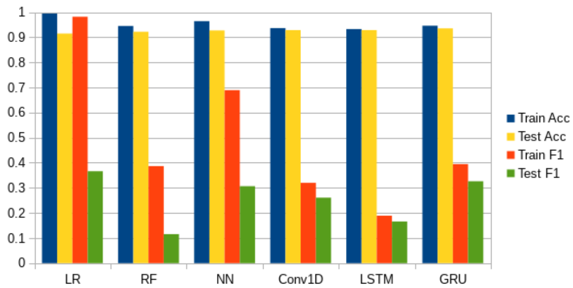
(a) top-10-code

(b) top-10-cat

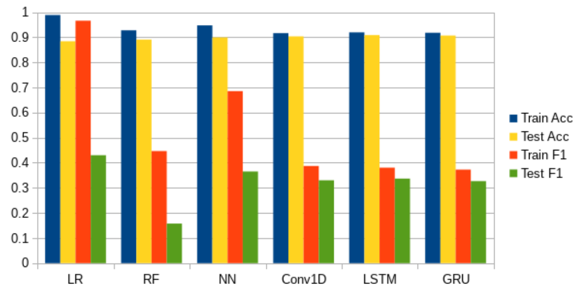
Figure 4: Model Performance Top 10

For *top-10-code* and *top-10-cat*, GRU generated the best f1 results (at 0.6957 and 0.7233, respectively). In general, *top-10-cat* generated slightly better results than *top-10-code*. This makes sense because 1) we have more data per labels in *top-10-cat* and 2) the labels are less specific (the differences between labels are larger). The baseline models (Logistic Regression and Random Forest) seems to overfit the data (which can be observed from the almost 100% training results but bad test results). Feed Forward NN is not overfitting, but the results are comparable to the baseline models. Conv1D produced even better results (than NN), but there are more significant improvement with LSTM and GRU (reaching around 70% f1). This signifies that our LSTM and GRU model was able to extract information from the sequence of words, otherwise lost in non-sequential feature extraction, thereby improving the f1 and also the accuracy results.

For *top-50-code* and *top-50-cat*, Logistic Regression generated the best f1 result (at 0.3662 and 0.4301, respectively). Similar to *top10*, *top-50-cat* generated slightly better results than *top-50-code*. The baseline models (Logistic Regression and Random Forest) also overfit here. However, the models that used sequential feature extraction did not produce



(a) top-50-code



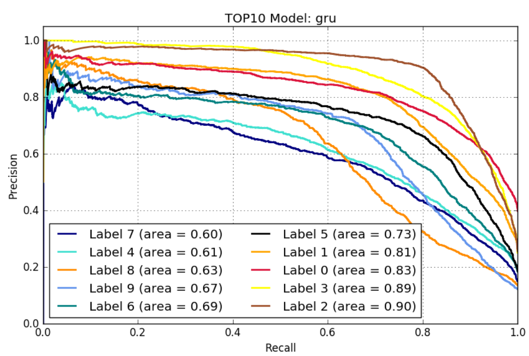
(b) top-50-cat

Figure 5: Model Performance Top 50

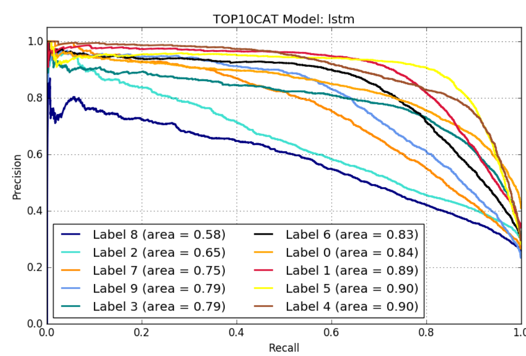
better results (in comparison with *top10*).

4.3.2 Precision-Recall Curve

Table 6 shows the average of overall precision performance of our selected best performance models for GRU, LSTM and Convolution 1D. From this table, we can see that GRU generated the best precision results for codes (*top-10-code*, *top-50-code*, and *top-50-code (top 10)*), LSTM generated the best precision results for categories (*top-10-cat*, *top-50-cat* and *top-50-cat (first 10)*). Figure 6 shows precision-recall curve for the best performed models for each labels in *top-10-codes* and *top-10-cat*. The detailed class-wise precision-recall curve for the selected models are presented in Appendix H



(a) top-10-code



(b) top-10-cat

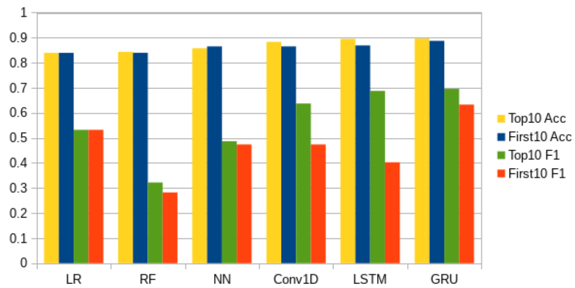
Figure 6: Class-wise Precision-Recall Curve for Top 10 and Top 10 Categories

Model	Top 10 Cod.	Top 10 Cat.	Top 50 Cod.	Top 50 Cat.	Top 50 Cod. (first 10)	Top 50 Cat. (first 10)
LSTM	0.7243	0.7915	0.3715	0.4929	0.7571	0.8426
GRU	0.7362	0.7849	0.4518	0.4792	0.7949	0.8425
Conv1D	0.6719	0.7293	0.3757	0.4565	0.7424	0.8269

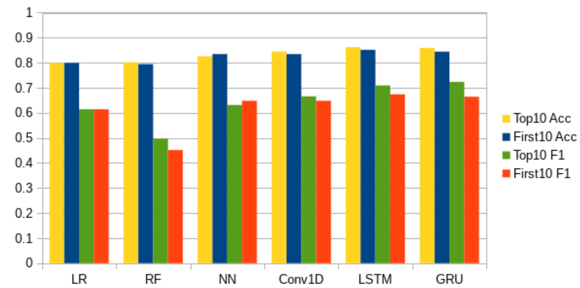
Table 6: Average Precision Performance

4.3.3 top-50-code and top-50-cat models

Figure 7 shows a comparison between the *top-10* results and the corresponding label results (first 10) in *top-50*. For models between ICD9 codes, the first-10 results are different from the top-10 results, implying that the *top-50-code* model is unable to completely cover the *top-10-code* model capability. For models between ICD9 categories, the first-10 results are similar to the top-10 results, implying that *top-50-cat* model is somewhat able to cover the *top-10-cat* model capability.



(a) top-10-code and top-50-code (first 10)



(b) top-10-cat and top-50-cat (first 10)

Figure 7: Top 10 vs Top 50 (first 10)

4.3.4 Results Comparison

Prakash and Zhao[5] use bag-of-words from discharge notes and Condensed Memory Neural Network (C-MemNN) to tackle the similar problem as we do. They tested their algorithm with top 50 and top 100 labels under metrics such as macro AUC, average precision over the top five predictions, and hamming loss. In order to compare our work with theirs, We calculated the same metrics for our best performed models for top 50 codes, and the results are listed in Table 7. From this comparison, we can see that our work outperformed on macro AUC and top 5 precision for top 50 codes. Their hamming loss is significantly better than ours. We believe that our word2vec representation plus memory network will make further progress on this problem.

Model	AUC (macro)	Precision @5	Hamming Loss
C-MemNN (Prakash)	0.833	0.42	**0.01**
GRU (our)	**0.8599**	**0.8109**	0.0645
LSTM (our)	0.8298	0.8054	0.0714
Conv1D (our)	0.8302	0.7998	0.0714

Table 7: Performance comparison with reference[5]

5 Conclusion

In this study, we applied deep learning based natural language processing frameworks to automatically assign clinical ICD-9 codes from free-text clinical notes. We compared different models and feature extraction methods, effectively establishing a strong benchmark for NLP in the MIMIC III dataset. We designed and produced deep learning models for predicting the top 10 ICD9 codes and categories which were better than our baseline models (best F1 results: 69.57% GRU to 53.20% Logistic Regression, and 72.33% GRU to 63.13% Feed Forward Neural Network, respectively). Top 50 ICD9 codes and categories results were not outperformed to our baseline (32.63% GRU to 36.62% Logistic Regression, and 33.67% GRU to 36.51% Feed Forward Neural Network, respectively). Even though our best model performance is around 70% F1, we hope our implementation and evaluation of the current state-of-the-art algorithms can serve as a baseline for further research on this topic.

6 Future Work

Our current models for top 50 ICD9 codes and categories were not as successful. This can be because our current model design lacks ”capability” in effectively distinguishing between 50 different labels. To improve our model capability, we can increase our neural network neurons (node count and layers), which greatly increases computation requirements / training time. Another approach is to produce 5 *top-10* models in parallel (each model predicting 10 labels), thereby making our *top-50* models have the same model capability as our *top-10* models, but also increasing computation requirements / training time by 5 times. Figure 8 shows the result of trying this approach for Conv1D using the *top-50-code* dataset. The parallel approach did improve F1 from 0.2609 to 0.3879, but to obtain *top-10-code*-level F1(0.6373), more samples (occurrences) for labels 11 to 50 is needed.

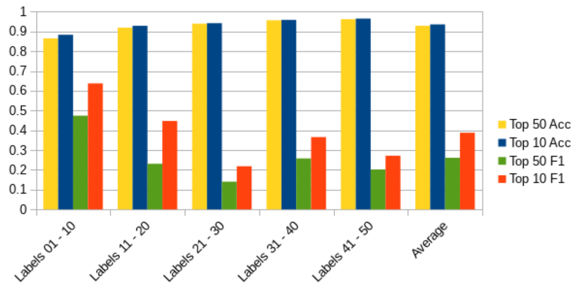


Figure 8: Parallel Top 10 vs Top 50

Our custom *word2vec* model used CBOW. We didn't have the chance to try skip-gram (although the pre-trained *word2vec* are skip-gram based). Some previous studies say that skip-gram outperforms CBOW in biomedical domain tasks[18]. Therefore in future work, we would like to see how the different *word2vec* parameters affect our ICD9 code or category classifier.

Further research on what words affect the probability of a prediction the most in our baseline models could improve our understanding of the relationship between symptoms and diagnosis. This could change our preprocessing and feature extraction methods and ultimately improve our deep learning models.

As discussed in Section 4.3.4, we are also interested in exploring more on memory networks and combining it with our input features.

References

1. Ashly D Black, Josip Car, Claudia Pagliari, Chantelle Anandan, Kathrin Cresswell, Tomislav Bokun, Brian McKinstry, Rob Procter, Azeem Majeed, and Aziz Sheikh. The impact of ehealth on the quality and safety of health care: a systematic overview. *PLoS Med*, 8(1):e1000387, 2011.

2. Edward Choi, Mohammad Taha Bahadori, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. *arXiv preprint arXiv:1511.05942*, 2015.

3. Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzell. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

4. Ping Chen, Araly Barrera, and Chris Rhodes. Semantic analysis of free text and its application on automatically assigning icd-9-cm codes to patient records. In *Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on*, pages 68–74. IEEE, 2010.

5. Aaditya Prakash, Siyuan Zhao, Sadid A Hasan, Vivek Datla, Kathy Lee, Ashequl Qadir, Joey Liu, and Oladimeji Farri. Condensed memory networks for clinical diagnostic inferencing. *arXiv preprint arXiv:1612.01848*, 2016.

6. F Martin-Sanchez, K Verspoor, et al. Big data in medicine is driving big changes. *Yearb Med Inform*, 9(1):14–20, 2014.

7. John P Pestian, Christopher Brew, Paweł Matykiewicz, Dj J Hovermale, Neil Johnson, K Bretonnel Cohen, and Włodzisław Duch. A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 97–104. Association for Computational Linguistics, 2007.

8. MBA Ira Goldstein and MLS Anna Arzumtsyan. Three approaches to automatic assignment of icd-9-cm codes to radiology reports. 2007.

9. Laritza M Rodriguez and Dina Demner Fushman. Automatic classification of structured product labels for pregnancy risk drug categories, a machine learning approach. In *AMIA Annual Symposium Proceedings*, volume 2015, page 1093. American Medical Informatics Association, 2015.

10. Ben J Marafino, Jason M Davies, Naomi S Bardach, Mitzi L Dean, R Adams Dudley, and John Boscardin. N-gram support vector machines for scalable procedure and diagnosis classification, with applications to clinical free text data from the intensive care unit. *Journal of the American Medical Informatics Association*, 21(5):871–875, 2014.

11. Vijay Garla, Caroline Taylor, and Cynthia Brandt. Semi-supervised clinical text classification with laplacian svms: an application to cancer case management. *Journal of biomedical informatics*, 46(5):869–875, 2013.

12. Adam Wright, Allison B McCoy, Stanislav Henkin, Abhivyakti Kale, and Dean F Sittig. Use of a support vector machine for categorizing free-text notes: assessment of accuracy across two institutions. *Journal of the American Medical Informatics Association*, 20(5):887–890, 2013.

13. Adler Perotte, Rimma Pivovarov, Karthik Natarajan, Nicole Weiskopf, Frank Wood, and Noémie Elhadad. Diagnosis code assignment: models and evaluation metrics. *Journal of the American Medical Informatics Association*, 21(2):231–237, 2014.

14. Priyanka Nigam. Applying deep learning to icd-9 multi-label classification from medical records. pages 1–8.

15. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

16. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

17. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

18. Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. How to Train Good Word Embeddings for Biomedical NLP. *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 166–174, 2016.

19. Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3), 2006.

20. Chuan Yu Foo Yifan Mai Caroline Suen Adam Coates Andrew Maas Awni Hannun Brody Huval Tao Wang Sameep Tandon Andrew Ng, Jiquan Ngiam. UFLDL Tutorial. http://ufldl.stanford.edu/tutorial/supervised/FeatureExtractionUsingConvolution/, 2008. [Online; accessed 18-April-2017].

21. Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 247–256. Association for Computational Linguistics, 2011.

22. Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 373–374. ACM, 2014.

23. Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

24. Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2014.

Appendices

A Model Performance for Top 10 Label Codes

	Training				Test			
Model	Precision	Recall	Accuracy	F1	Precision	Recall	Accuracy	F1
Logistic Regression	0.9564	0.9440	0.9786	0.9501	0.5801	0.4934	0.8392	0.5320
Random Forest	0.9989	0.6988	0.9501	0.8086	0.7573	0.2340	0.8432	0.3219
Feed Forward NN	0.7768	0.5041	0.8879	0.5763	0.6703	0.4193	0.8575	0.4868
Conv1D	0.8312	0.6713	0.9165	0.7371	0.7408	0.5687	0.8832	0.6373
LSTM RNN	0.8106	0.6971	0.9154	0.7445	0.7574	0.6380	0.8950	0.6874
GRU RNN	0.7936	0.6971	0.9126	0.7397	0.7502	0.6519	0.8967	0.6957

Table 8: Model Performance for *top-10-code*

B Model Performance for Top 50 Codes

	Training				Test			
Model	Precision	Recall	Accuracy	F1	Precision	Recall	Accuracy	F1
Logistic Regression	0.9863	0.9768	0.9945	0.9815	0.4372	0.3213	0.9148	0.3662
Random Forest	0.9985	0.2852	0.9451	0.3866	0.5377	0.0953	0.9220	0.1155
Feed Forward NN	0.9636	0.5542	0.9640	0.6892	0.5773	0.2335	0.9271	0.3067
Conv1D	0.6085 [a]	0.2663	0.9365	0.3200 [a]	0.4792 [a]	0.2169	0.9286	0.2609 [a]
LSTM RNN	0.3526 [a]	0.1642	0.9325	0.1891 [a]	0.4022 [a]	0.1445	0.9286	0.1659 [a]
GRU RNN	0.6539 [a]	0.3433	0.9460	0.3947 [a]	0.5592 [a]	0.2782	0.9354	0.3263 [a]

Table 9: Model Performance for *top-50-code*

	Training				Test			
Model	Precision	Recall	Accuracy	F1	Precision	Recall	Accuracy	F1
Logistic Regression	0.9564	0.9440	0.9786	0.9501	0.5801	0.4934	0.8392	0.5320
Random Forest	0.9946	0.4937	0.9110	0.6305	0.7869	0.2009	0.8395	0.2822
Feed Forward NN	0.9347	0.8037	0.9580	0.8589	0.7674	0.5837	0.9062	0.6486
Conv1D	0.7708	0.4673	0.8858	0.5377	0.6784	0.4109	0.8650	0.4739
LSTM RNN	0.6204[a]	0.3829	0.8805	0.4348[a]	0.5748	0.3526	0.8688	0.4025[a]
GRU RNN	0.8351	0.6474	0.9168	0.7181	0.7520	0.5618	0.8871	0.6328

Table 10: Model Performance for *top-50-code* (first 10)

[a]result contained nan. Computed by replacing nan with zero.

C Model Performance for Top 10 Label Categories

	Training				Test			
Model	Precision	Recall	Accuracy	F1	Precision	Recall	Accuracy	F1
Logistic Regression	0.9437	0.9309	0.9652	0.9372	0.6458	0.5856	0.7994	0.6141
Random Forest	0.9983	0.8134	0.9500	0.8954	0.7653	0.3801	0.8019	0.4966
Feed Forward NN	0.7978	0.6575	0.8655	0.7147	0.7243	0.5689	0.8257	0.6313
Conv1D	0.8039	0.6637	0.8681	0.7128	0.7613	0.6126	0.8446	0.6657
LSTM RNN	0.8146	0.6807	0.8749	0.7343	0.7926	0.6536	0.8622	0.7090
GRU RNN	0.8150	0.7613	0.8909	0.7861	0.7580	0.6941	0.8588	0.7233

Table 11: Model Performance for *top-10-cat*

D Model Performance for Top 50 Label Categories

	Training				Test			
Model	Precision	Recall	Accuracy	F1	Precision	Recall	Accuracy	F1
Logistic Regression	0.9750	0.9572	0.9887	0.9659	0.4858	0.3894	0.8841	0.4301
Random Forest	0.9986	0.3294	0.9277	0.4465	0.6568	0.1142	0.8906	0.1576
Feed Forward NN	0.9613	0.5488	0.9473	0.6853	0.6298	0.2773	0.8992	0.3651
Conv1D	0.7428	0.3262	0.9163	0.3870	0.5635 [a]	0.2770	0.9035	0.3301 [a]
LSTM RNN	0.7117 [a]	0.3363	0.9194	0.3804 [a]	0.5869	0.2945	0.9087	0.3367 [a]
GRU RNN	0.6695 [a]	0.3227	0.9179	0.3726 [a]	0.5611 [a]	0.2809	0.9067	0.3266 [a]

Table 12: Model Performance for *top-50-cat*

	Training				Test			
Model	Precision	Recall	Accuracy	F1	Precision	Recall	Accuracy	F1
Logistic Regression	0.9437	0.9309	0.9652	0.9372	0.6458	0.5856	0.7994	0.6141
Random Forest	0.9937	0.6321	0.8999	0.7687	0.7877	0.3282	0.7944	0.4512
Feed Forward NN	0.7873	0.6425	0.8811	0.7031	0.7730	0.6267	0.8724	0.6878
Conv1D	0.7945	0.6652	0.8670	0.7142	0.7296	0.5979	0.8345	0.6481
LSTM RNN	0.7963	0.6863	0.8768	0.7213	0.7515	0.6362	0.8514	0.6738
GRU RNN	0.7901	0.6803	0.8729	0.7213	0.7382	0.6196	0.8442	0.6641

Table 13: Model Performance for *top-50-cat* (first 10)

[a]result contained nan. Computed by replacing nan with zero.

E Model Performance under Different Configurations (Non Sequential)

Model	Configuration	Feature	Precision	Recall	Accuracy	F1
Logistic Regression	Iter=25	tfidfv0	0.5290	0.4445	0.8232	0.4810
Logistic Regression	Iter=10	tfidfv1	0.5801	0.4934	0.8392	0.5320
Logistic Regression	Iter=50	word2vec_f100	0.5820	0.2682	0.8353	0.3393
Logistic Regression	Iter=100	word2vec_f300	0.6265	0.3388	0.8448	0.4191
Logistic Regression	Iter=100	word2vec_f600	0.6467	0.3914	0.8513	0.4749
Logistic Regression	Iter=100	word2vec_bio02	0.5938	0.2807	0.8340	0.3568
Logistic Regression	Iter=75	word2vec_bio30	0.6030	0.2921	0.8370	0.3706
Logistic Regression	Iter=100	doc2vec_f100	0.0243	0.0021	0.8015	0.0039
Logistic Regression	Iter=75	doc2vec_f300	0.0000	0.0000	0.8025	0.0000
Logistic Regression	Iter=50	doc2vec_f600	0.0000	0.0000	0.8025	0.0000
Random Forest	Depth=30	tfidfv0	0.7529	0.1881	0.8354	0.2676
Random Forest	Depth=30	tfidfv1	0.7573	0.2340	0.8432	0.3219
Random Forest	Depth=20	word2vec_f100	0.5281	0.1907	0.8233	0.2585
Random Forest	Depth=20	word2vec_f300	0.5471	0.1866	0.8237	0.2559
Random Forest	Depth=20	word2vec_f600	0.5296	0.1857	0.8228	0.2543
Random Forest	Depth=20	word2vec_bio02	0.5420	0.1794	0.8224	0.2480
Random Forest	Depth=20	word2vec_bio30	0.5283	0.1852	0.8225	0.2530
Random Forest	Depth=30	doc2vec_f100	0.2031	0.2040	0.6924	0.1437
Random Forest	Depth=20	doc2vec_f300	0.2122	0.2860	0.6387	0.1971
Random Forest	Depth=20	doc2vec_f600	0.2026	0.2044	0.7073	0.1705
Feed Forward NN	nn_model_1	tfidfv0	0.6975	0.3785	0.8542	0.4795
Feed Forward NN	nn_model_2	tfidfv1	0.6703	0.4193	0.8575	0.4868
Feed Forward NN	nn_model_2	word2vec_f100	0.5429	0.3505	0.8304	0.4116
Feed Forward NN	nn_model_2	word2vec_f300	0.5467	0.4627	0.8329	0.4929
Feed Forward NN	nn_model_2	word2vec_f600	0.5737	0.4972	0.8406	0.5285
Feed Forward NN	nn_model_2	word2vec_bio02	0.54264	0.277848	0.837748	0.340831
Feed Forward NN	nn_model_4	word2vec_bio30	0.5228	0.3856	0.8253	0.4375
Feed Forward NN	nn_model_2	doc2vec_f100	0.0000	0.0000	0.8025	0.0000
Feed Forward NN	nn_model_2	doc2vec_f300	0.0000	0.0000	0.8025	0.0000
Feed Forward NN	nn_model_2	doc2vec_f600	0.0000	0.0000	0.8025	0.0000

Table 14: Model Performance under Different Configurations (Non Sequential)

[a] result contained nan. Computed by replacing nan with zero.

F Model Performance under Different Configurations (Sequential)

Model	Configuration	Feature	Embedding Matrix	Precision	Recall	Accuracy	F1
Conv1D	conv1d_6	seqv0	word2vec_f100	0.7460	0.4626	0.8758	0.5489
Conv1D	conv1d_6	seqv0	word2vec_f300	0.7492	0.5103	0.8779	0.5846
Conv1D	conv1d_6	seqv0	word2vec_f600	0.7408	0.5687	0.8832	0.6373
Conv1D	conv1d_6	seqv0	word2vec_bio02	0.6424	0.4962	0.8642	0.5464
Conv1D	conv1d_6	seqv0	word2vec_bio30	0.7283	0.5217	0.8757	0.5888
Conv1D	conv1d_6	seqv1	word2vec_f100	0.7304	0.4851	0.8744	0.5565
Conv1D	conv1d_6	seqv1	word2vec_f300	0.7141	0.5443	0.8762	0.6074
Conv1D	conv1d_6	seqv1	word2vec_f600	0.7578	0.5081	0.8805	0.5972
Conv1D	conv1d_6	seqv1	word2vec_bio02	0.5635 [a]	0.2770	0.9035	0.3301 [a]
Conv1D	conv1d_6	seqv1	word2vec_bio30	0.6819	0.5323	0.8658	0.5771
RNN	rnn_2	seqv0	word2vec_f100	0.1772 [a]	0.0758	0.8067	0.08693 [a]
RNN	rnn_2	seqv0	word2vec_f300	0.2291 [a]	0.0476	0.8067	0.06774 [a]
RNN	rnn_2	seqv0	word2vec_f600	0.1110 [a]	0.0513	0.8052	0.0702 [a]
RNN	rnn_2	seqv0	word2vec_bio02	0.0000 [a]	0.0000	0.8025	0.0000 [a]
RNN	rnn_2	seqv0	word2vec_bio30	0.0000 [a]	0.0000	0.8025	0.0000 [a]
RNN	rnn_2	seqv1	word2vec_bio02	0.1150 [a]	0.0390	0.8045	0.0535 [a]
RNN	rnn_2	seqv1	word2vec_bio30	0.1087 [a]	0.0545	0.8025	0.0630 [a]
LSTM	lstm_1	seqv0	word2vec_f100	0.6857	0.4958	0.8749	0.5499
LSTM	lstm_1	seqv0	word2vec_f300	0.7577	0.6001	0.8922	0.6664
LSTM	lstm_1	seqv0	word2vec_f600	0.7287	0.6381	0.8901	0.6768
LSTM	lstm_1	seqv0	word2vec_bio02	0.7464	0.6381	0.8931	0.6831
LSTM	lstm_1	seqv0	word2vec_bio30	0.7529	0.6288	0.8939	0.6794
LSTM	lstm_1	seqv1	word2vec_f100	0.7275	0.6523	0.8910	0.6862
LSTM	lstm_1	seqv1	word2vec_f300	0.7566	0.6281	0.8948	0.6829
LSTM	lstm_1	seqv1	word2vec_f600	0.7549	0.5974	0.8908	0.6556
LSTM	lstm_1	seqv1	word2vec_bio02	0.7702	0.5984	0.8930	0.6654
LSTM	lstm_1	seqv1	word2vec_bio30	0.7574	0.6380	0.8950	0.6874
GRU	gru_4	seqv0	word2vec_f100	0.7456	0.6486	0.8953	0.6902
GRU	gru_4	seqv0	word2vec_f300	0.7502	0.6519	0.8967	0.6957
GRU	gru_4	seqv0	word2vec_f600	0.7395	0.6164	0.8893	0.6651
GRU	gru_4	seqv0	word2vec_bio02	0.7166	0.5572	0.8804	0.6128
GRU	gru_4	seqv0	word2vec_bio30	0.7577	0.5672	0.8868	0.6313
GRU	gru_4	seqv1	word2vec_f100	0.7475	0.6514	0.8955	0.6930
GRU	gru_4	seqv1	word2vec_f300	0.7758	0.6111	0.8967	0.6776
GRU	gru_4	seqv1	word2vec_f600	0.7557	0.6389	0.8955	0.6881
GRU	gru_4	seqv1	word2vec_bio02	0.7228	0.5773	0.8815	0.6273
GRU	gru_4	seqv1	word2vec_bio30	0.7404	0.6277	0.8907	0.6769

Table 15: Model Performance under Different Configurations (Sequential)

[a] result contained nan. Computed by replacing nan with zero.

G Best Performance Model Architectures

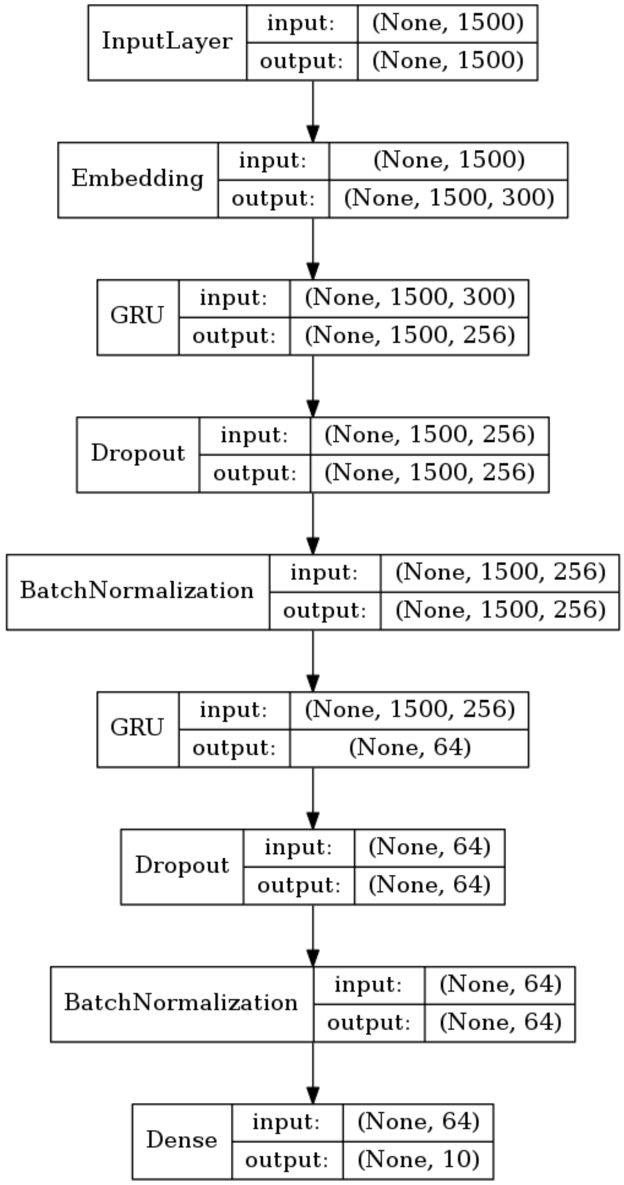


Figure 9: Best GRU Model Architecture for Top 10 Codes, Top 50 Codes, Top 10 Categories, and Top 50 Categories

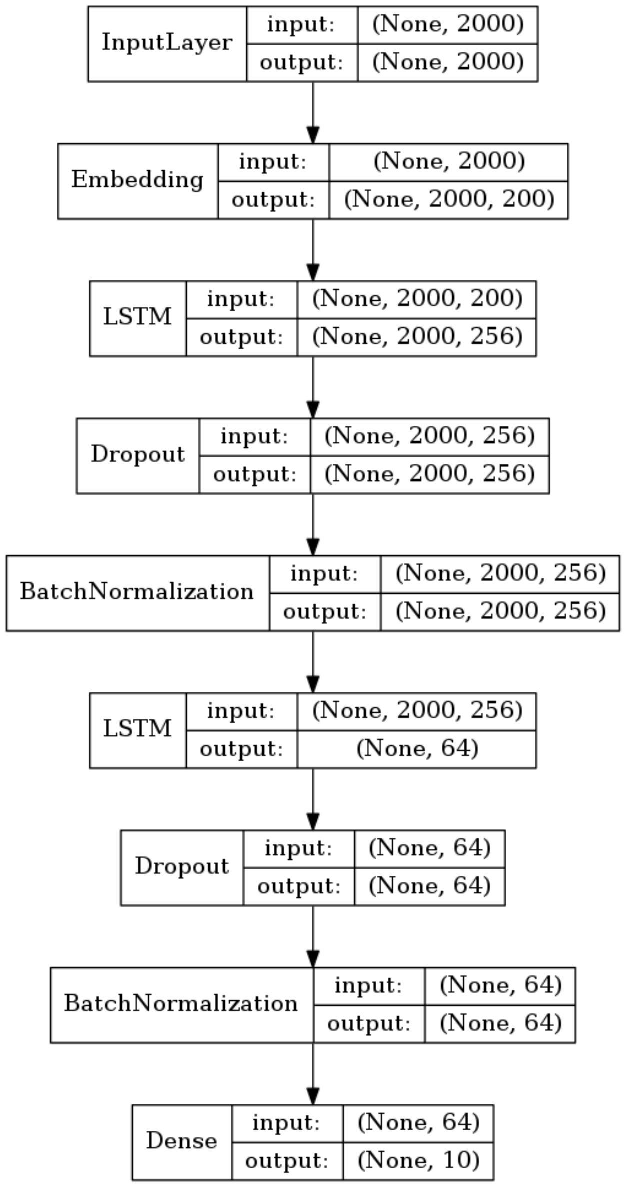


Figure 10: Best LSTM Model Architecture for Top 10 Codes, Top 50 Codes, Top 10 Categories, and Top 50 Categories

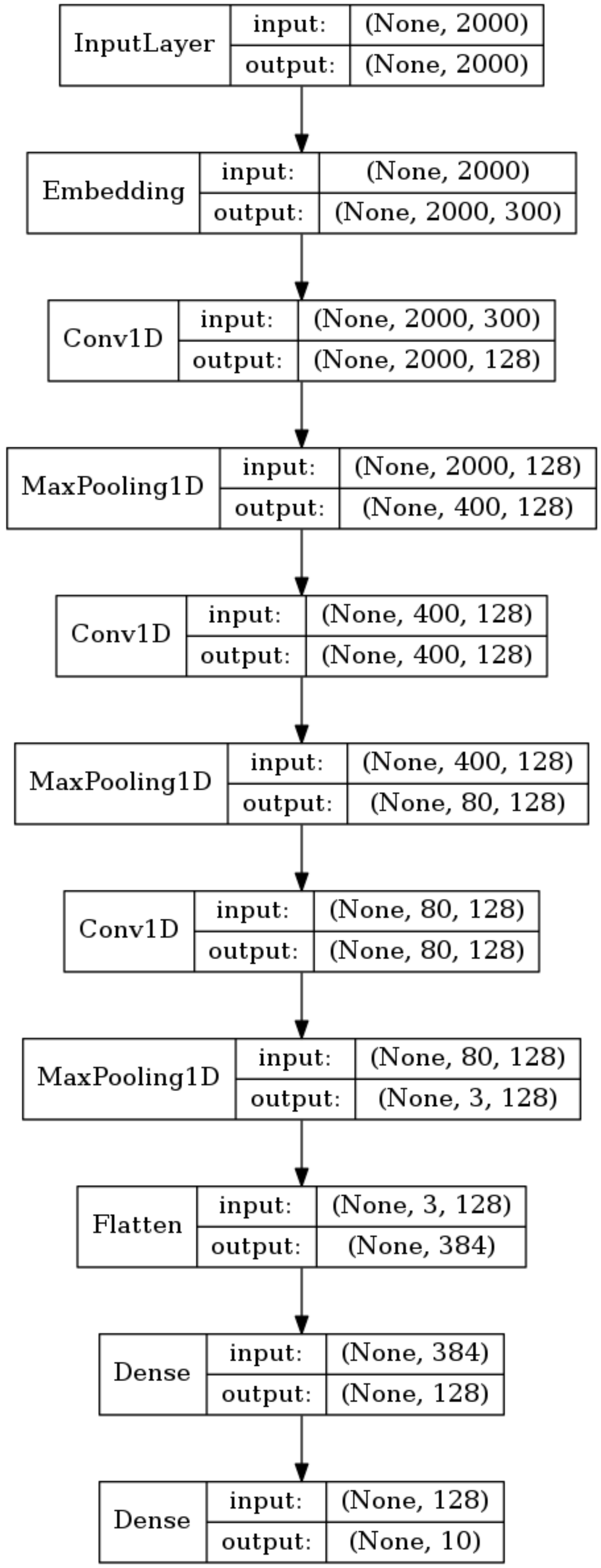


Figure 11: Best Convolution 1D Model Architecture for Top 10 Codes and Top 10 Categories

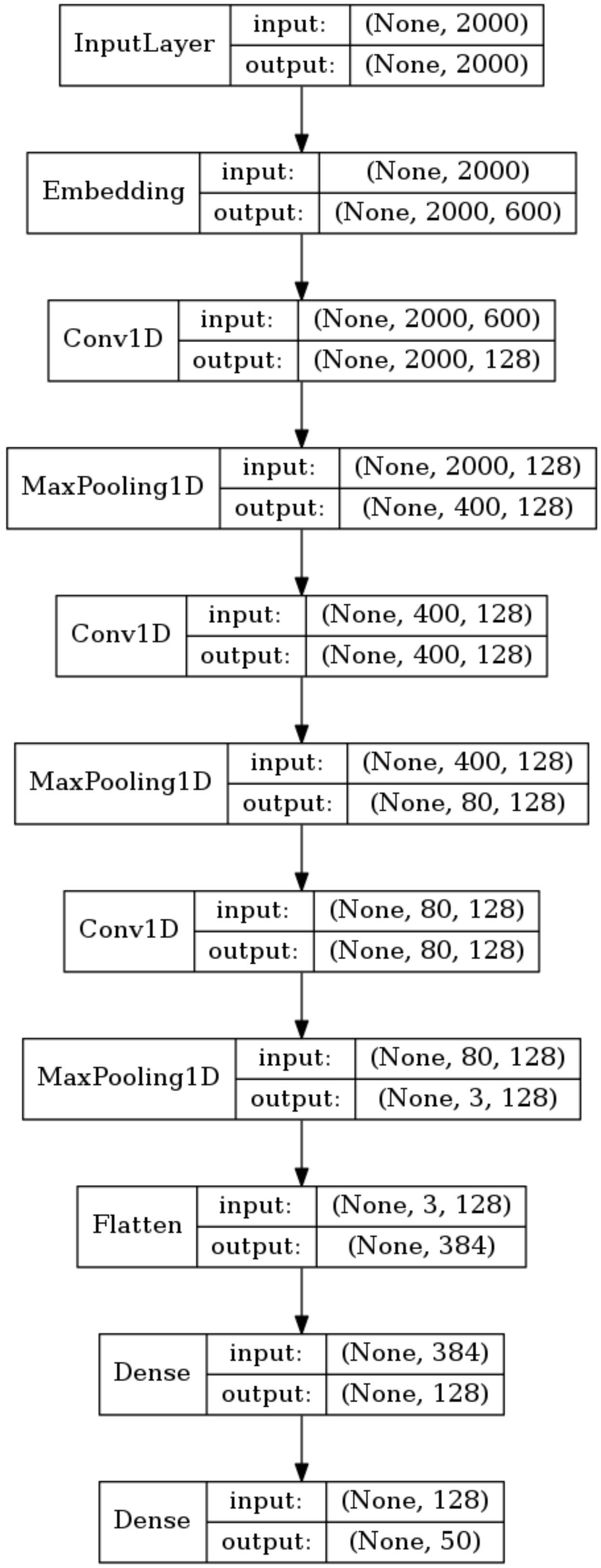


Figure 12: Best Convolution 1D Model Architecture for Top 50 Codes and Top 50 Categories

H Best Performance Models (LSTM, GRU, Conv1D) Precision-Recall Curve

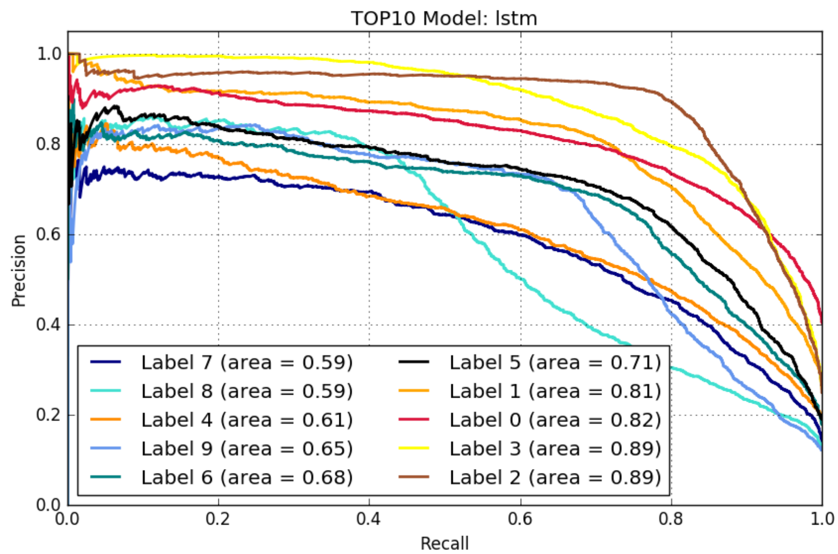


Figure 13: Precision-Recall Curve for Top 10 Codes under LSTM Model

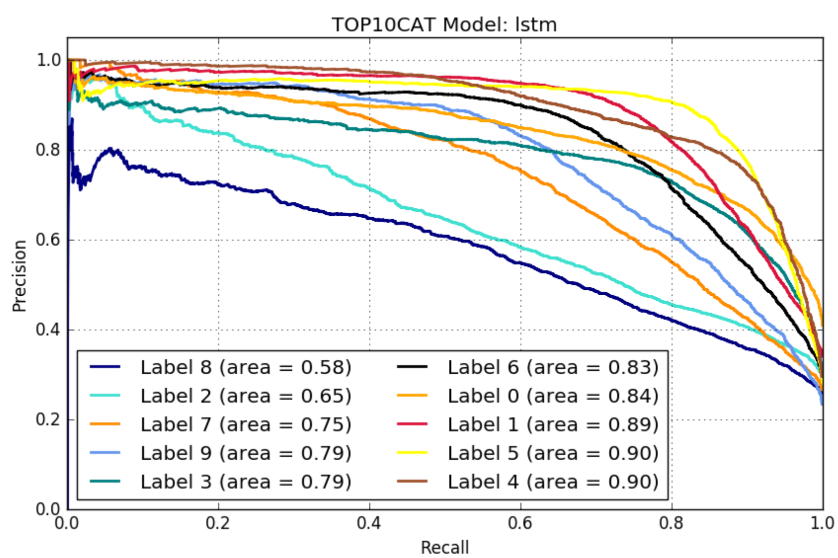


Figure 14: Precision-Recall Curve for Top 10 Categories under LSTM Model

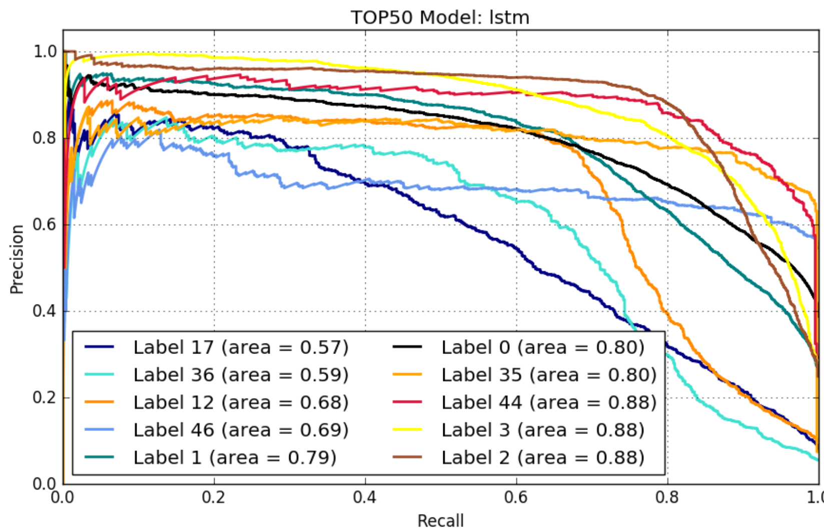


Figure 15: Precision-Recall Curve for Top 50 Codes under LSTM Model

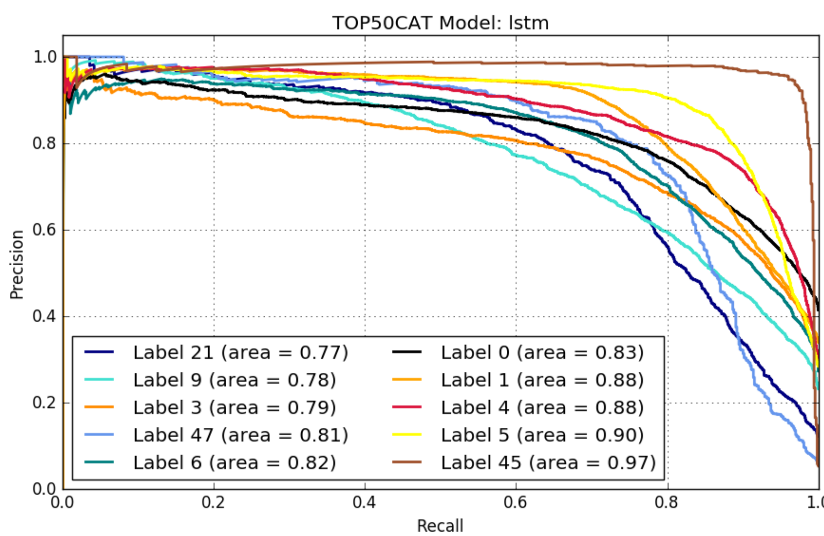


Figure 16: Precision-Recall Curve for Top 50 Categories under LSTM Model

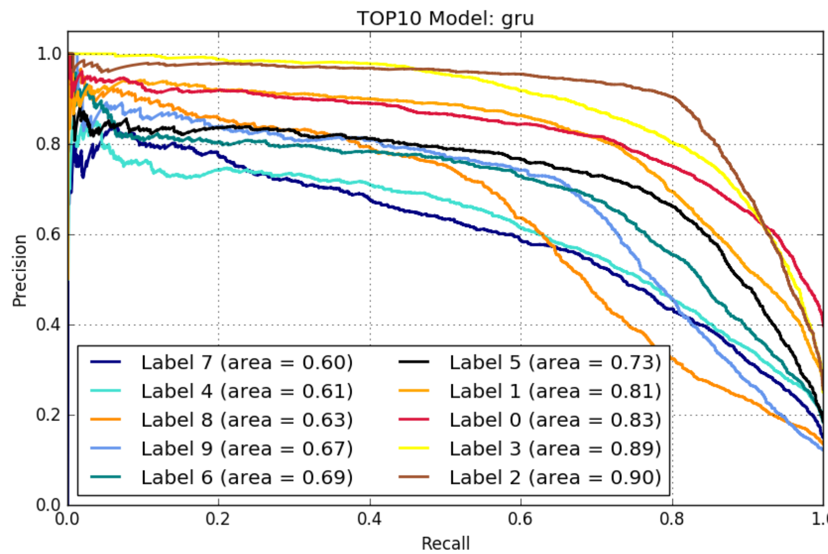


Figure 17: Precision-Recall Curve for Top 10 Codes under GRU Model

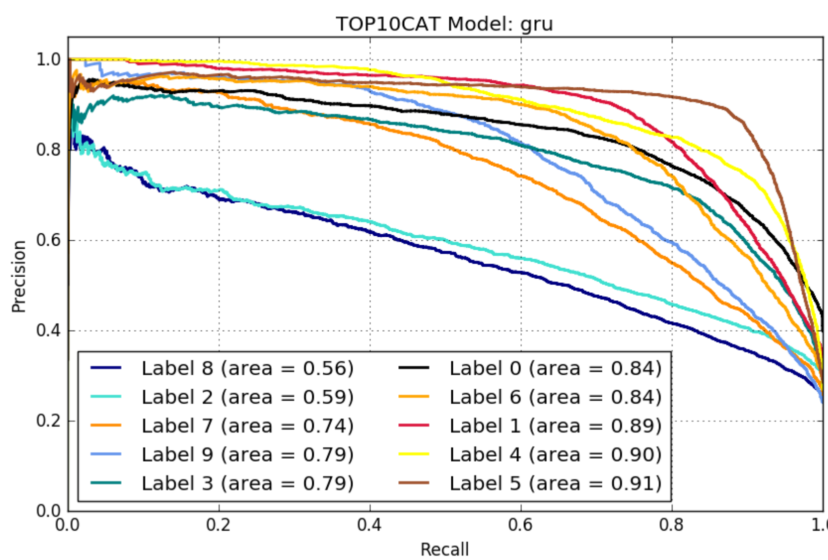


Figure 18: Precision-Recall Curve for Top 10 Categories under GRU Model

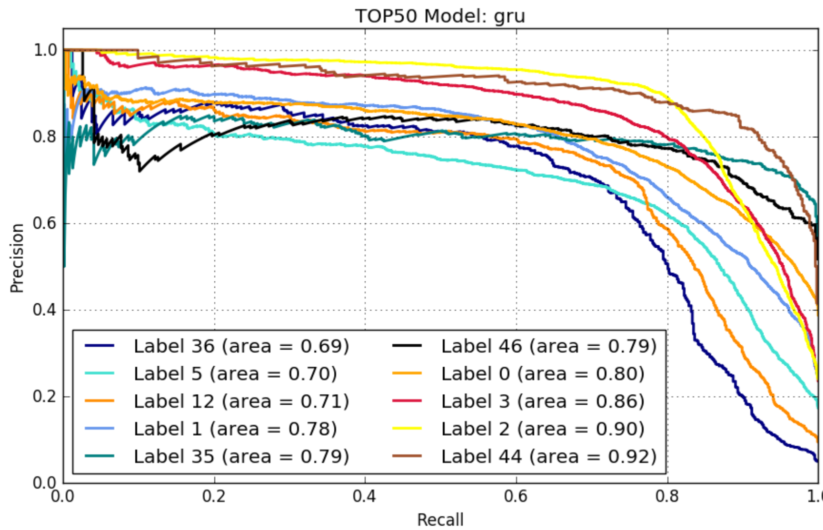


Figure 19: Precision-Recall Curve for Top 50 Codes under GRU Model

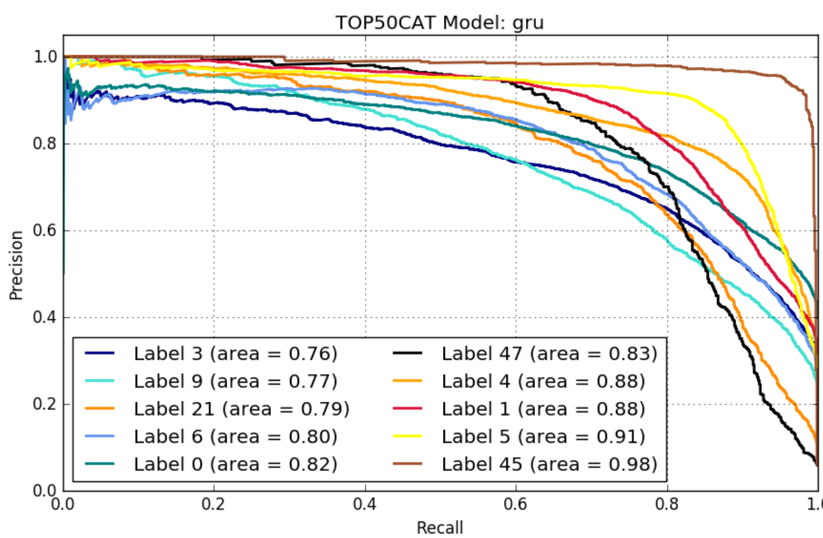


Figure 20: Precision-Recall Curve for Top 50 Categories under GRU Model

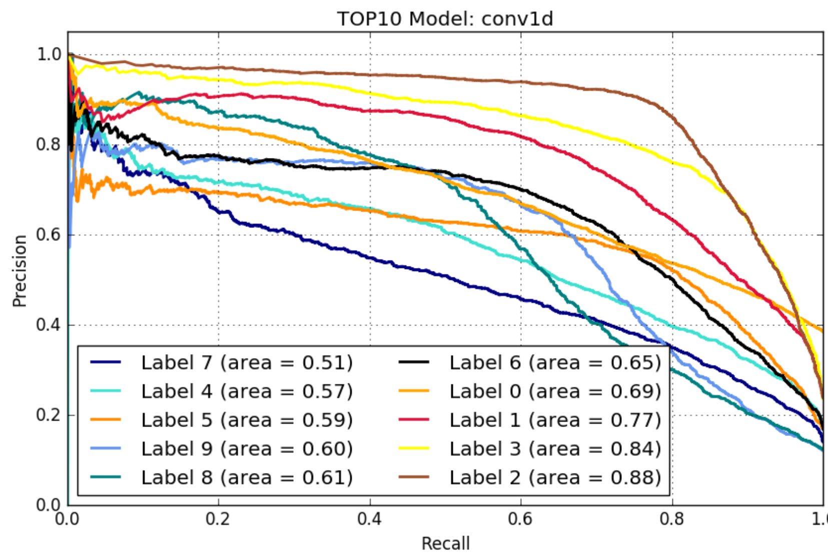


Figure 21: Precision-Recall Curve for Top 10 Codes under Convolution 1D Model

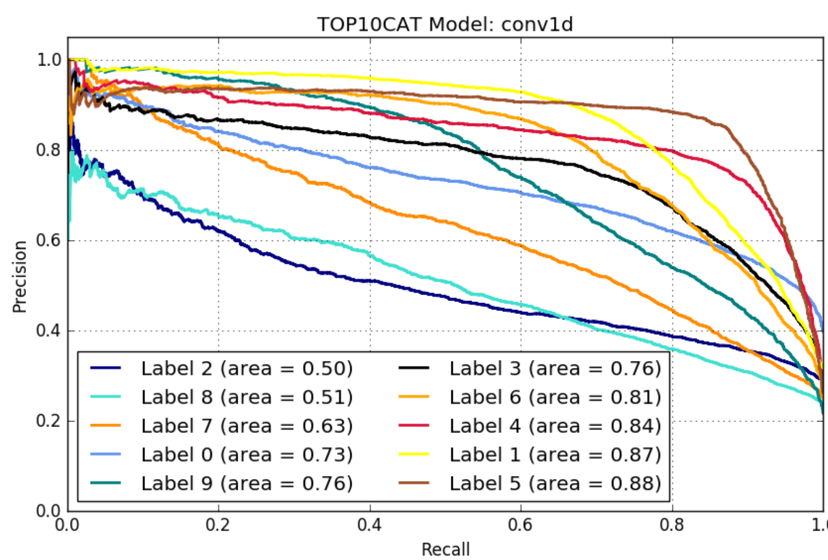


Figure 22: Precision-Recall Curve for Top 10 Categories under Convolution 1D Model

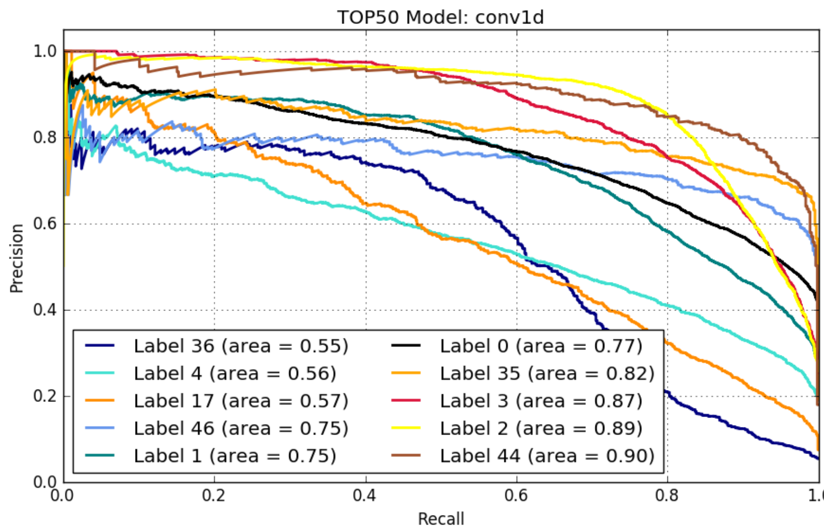


Figure 23: Precision-Recall Curve for Top 50 Codes under Convolution 1D Model

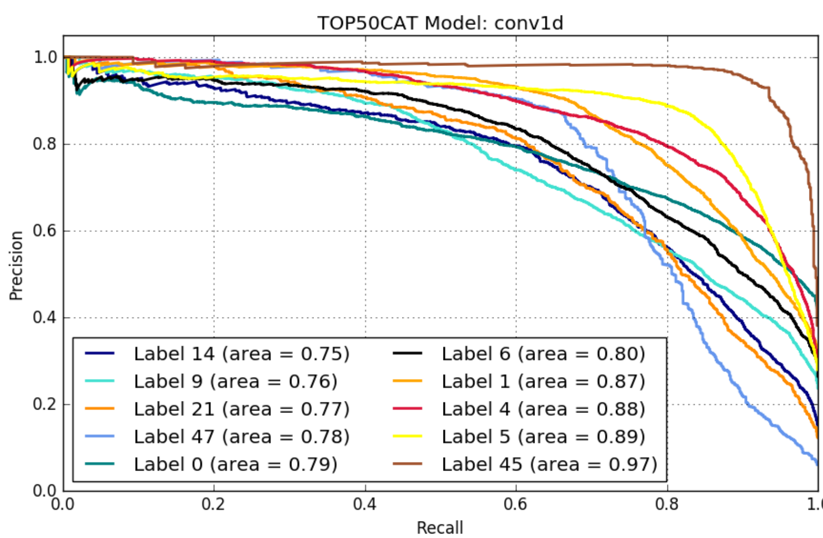


Figure 24: Precision-Recall Curve for Top 50 Categories under Convolution 1D Model