CS598 DL4H in Spring 2023 - Automated ICD-9 Coding via A Deep Learning Approach

Ryan Fogle
rsfogle2@illinois.edu

Group ID: 203
Paper ID: 187
Presentation link: https://youtu.be/Wn_x4QYvTjM
Code link: https://github.com/RynoXLI/DL4H-Automated-ICD-9-Coding

1 Introduction

ICD-9 coding is a time-consuming task that requires a specialized skill set to provide accurate ICD-9 codings. Using the discharge summaries and ICD-9 diagnostic codes provided in the MIMIC-III dataset, the paper "Automated ICD-9 Coding via A Deep Learning Approach" introduces a multi-label classification problem. The authors of the paper introduce a deep learning model called DeepLabeler which incorporates two models: a Word2Vec model (Mikolov et al., 2013) and a Doc2Vec model (Le and Mikolov, 2014). The paper claims DeepLabeler performs better (via an F1 score) than a traditional natural language processing model like a support vector machine (Li et al., 2019).

2 Scope of reproducibility

The original paper proposes that their model produces a better micro-f1 score than a traditional linear SVM model. By reproducing a linear SVM model and creating the DeepLabeler architecture, we can evaluate whether non-linear representations can better describe automatic ICD9 coding.

The paper additionally proposes adding a Doc2Vec model to the model architecture increases the micro-f1 score. The author reasoned that the Doc2Vec model captures the global context of the discharge summaries. By removing the Doc2Vec model, we can test whether the Doc2Vec increases the micro-f1 score.

Additionally, this paper did not explore the possibility of replacing the Word2Vec layer with an embedding layer. In this report, we will transfer the embeddings of the Word2Vec model to an embedding layer and measure the micro-f1 score. We will be able to see if letting the model change the embeddings will increase the micro-f1 score.

The micro-f1 score is described as the following:

$$MiP = \frac{\sum_{m=1}^{M}\sum_{i=1}^{N} y_i^m \hat{y}_i^m}{\sum_{m=1}^{M}\sum_{i=1}^{N} \hat{y}_i^m} \tag{1}$$

$$MiR = \frac{\sum_{m=1}^{M}\sum_{i=1}^{N} y_i^m \hat{y}_i^m}{\sum_{m=1}^{M}\sum_{i=1}^{N} y_i^m} \tag{2}$$

$$\text{Micro F1} = \frac{2 \times MiP \times MiR}{MiP + MiR} \tag{3}$$

2.1 Addressed claims from the original paper

- The DeepLabeler architecture produces a better micro-f1 score than a linear SVM.
- The DeepLabeler architecture produces a higher micro-f1 score when a Doc2Vec model is introduced.

2.2 Ablations

- Replace the Word2Vec model with an embedding layer and report the f1-score.

3 Methodology

For this paper, a new code repository was created because the original authors did not provide one. The model descriptions seen in this paper are my interpretations of how the original authors implemented DeepLabeler. For this report, I used a 5600X Ryzen processor, 32GB of RAM, and an Nvidia 3080 with 10GB of VRAM.

3.1 Model descriptions

Instead of using the traditional approach of bag-of-words, TF-IDF, and an SVM, DeepLabeler uses a deep learning approach. DeepLabeler pre-trains two models: a Word2Vec model and a Doc2Vec model on the patient discharge summaries. Using both models, for each record a document-level vector is created and word vectors are created for each word in the summary. The document vector is put

through one fully connected layer, the word vectors are arranged in a matrix, and a convolutional layer is applied followed by a max pool. Both of these outputs are concatenated together and put through a sigmoid layer for the final classification. The total number of trainable parameters was 343,692. A high-level overview of the model architecture can be seen in Figure 1.

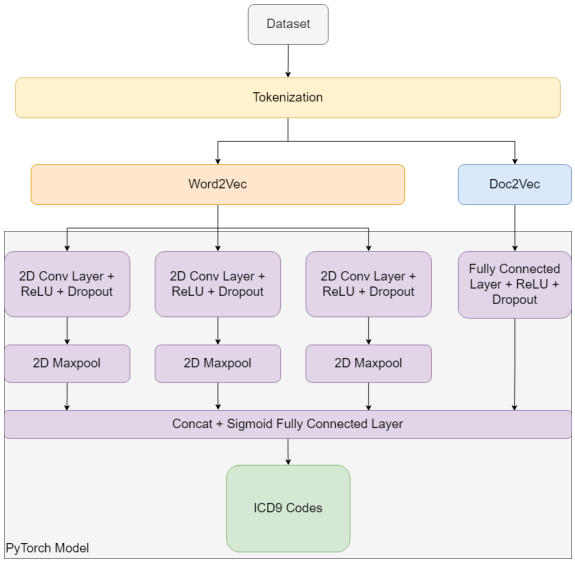

Figure 1: DeepLabeler Architecture

The model architecture without the Doc2Vec model can be seen in Figure 2. Additionally, the model architecture for this report's ablation study can be seen in Figure 2 as well, except replace the Word2Vec model with an embedding layer.

For all DeepLabeler-based models, Binary Cross Entropy loss was used along with adaptive moment estimation.

For the linear SVM model, a more traditional NLP approach was taken. A bag-of-words matrix was formed and then transformed using TF-IDF. In order to create a multi-label output, each ICD9 code had its own model trained following the One-vs-Rest ideology. Finally, cross-validation was used to produce threshold probabilities for each label.

3.2 Data descriptions

The dataset used for this paper is MIMIC-III. To obtain the dataset you must complete the CITI Program training, and have downloaded the MIMIC-III data from Physionet website (Johnson, 2022). A high-level overview of the preprocessed and tokenized data can be viewed in Table 1.

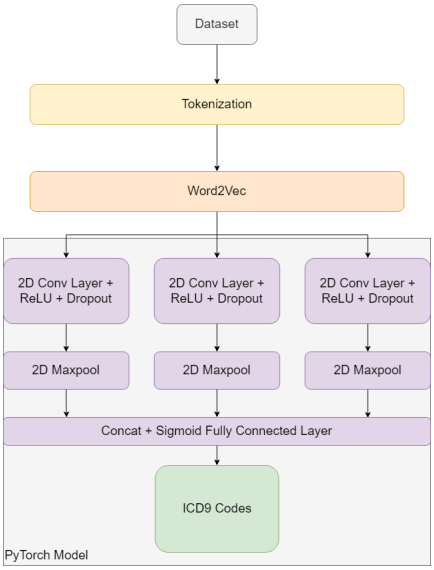

Figure 2: DeepLabeler without Doc2Vec Architecture

Data Descriptions	
Data Statistic	Value
No. of Discharge Summaries	52,622
No. of ICD9 Codes	1,292
Ave. Word Length	1,429
Std. Word Length	734
Max. Word Length	8,148
Min. Word Length	28

Table 1: Data descriptions of the Discharge Summaries and their corresponding ICD9 Codes

3.3 Hyperparameters

The hyperparameters given by the paper are seen in table 2, there were no other hyperparameters given. The learning rate used for this report was 10^{-3}, and each DeepLabeler model was trained for 5 epochs. The learning rate was based on empirical evidence of a stable convergence of loss, and epochs of 5 was chosen for quick training and convergence.

For the SVM section, a minimum document frequency was set to 0.1% for the TF-IDF algorithm. The loss function of the SVM was set to hinge, and the "True" class weight was set to 10 while "False" class set to 1.

3.4 Implementation

For this report, I created a new repository (DL4H-Automated-ICD-9-Coding). The Doc2Vec and Word2Vec models were built using the Gensim python library (Řehůřek and Sojka, 2010). DeepLabeler architecture was built using Pytorch 2.0 (Paszke et al., 2019). The SVM model was built us-

Hyperparameters	
Convolutions	CNN Part
Word embedding size	100
Maximum Length	700
Kernel Size	3, 4, 5
Input channels	1
Output channels	64, 64, 64
dropout rate	0.75
Doc2Vec Part	
Document embedding size	128
Fully connected layer size	64
Dropout rate	0.75

Table 2: DeepLabeler Hyperparameters given by Paper

ing the scikit-learn Python library (Pedregosa et al., 2011).

3.5 Computational requirements

For this report, I split the code into seven different parts. A high-level overview of the computation times can be viewed in Table 3.

Computation Times (Minutes:Seconds)		
Code Section	Time per Epoch	Total Time
Data Preparation	N/A	1:58
SVM	N/A	27:52
Word2Vec	0:43	6:42
Doc2Vec	0:23	15:51
DeepLabeler	2:27	13:01
DeepLabeler w/o Doc2Vec	2:34	13:37
DeepLabeler w/ Embedding	2:12	12:03
Total Time	N/A	91:04

Table 3: Computation times in minutes and seconds

All DeepLabeler models used the GPU, although the Doc2Vec, Word2Vec, SVM, and data preparation scripts only used the CPU.

4 Results

The results of this report do not support all of the claims made in the paper. The main premise of the paper is that by adding a deep learning architecture we can increase the micro-f1 score, my findings do not support that claim. Although, this report does support the claim that by adding the Doc2Vec vectors, the micro-f1 score increases. Additionally, I find that replacing the Word2Vec model with an embedding layer reduces the micro-f1 score. A high-level overview of the results can be viewed in table 4.

Model Metrics			
Model Type w/ Threshold	Precision	Recall	F1-Score
SVM @ 0.2	0.48	0.46	0.47
DeepLabeler @ 0.4	0.22	0.29	0.25
DeepLabeler w/o Doc2Vec @ 0.4	0.27	0.17	0.21
DeepLabeler w/ Embedding @ 0.4	0.30	0.15	0.20

Table 4: Model Evaluation Metrics

An analysis of the micro precision, recall, and f1 scores for each threshold and model can be viewed in the following figures: 3 (SVM), 4 (DeepLabler), 5 (DeepLabeler without D2V), and 6 (DeepLabeler with Embeddings).

4.1 Result 1

As you can see in Figure 3 and Figure 4, the linear SVM model outperforms the DeepLabeler architecture on all metrics. This is contrary to the original claim that the micro-f1 score would increase with the proposed DeepLabeler architecture. You will also notice that if we had simply gone with the threshold at 0.5 for the SVM, then we would have reduced our f1-score to 0.38. The metrics for the SVM in this report are greatly improved from the SVM metrics included in the original paper, although the metrics for the DeepLabeler included in this report are worse than the authors' metrics.

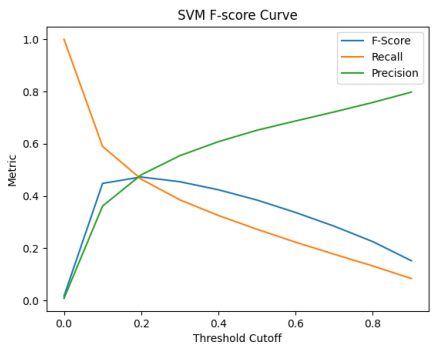

Figure 3: SVM Micro-F1 Score Curve

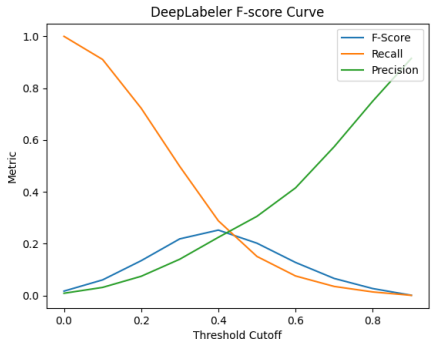

Figure 4: DeepLabeler Micro-F1 Score Curve

4.2 Result 2

From Figure 5 and Figure 4, we can see that the modified DeepLabeler architecture performed worse with recall and f1, although it performed better with precision. This result is slightly different than the original paper, although the conclusion is the same as the paper: the DeepLabeler architecture micro-f1 score decreases when removing the Doc2Vec model. You will also notice that the precision reduces to zero at high thresholds with the new model whereas with the original DeepLabeler that did not occur.

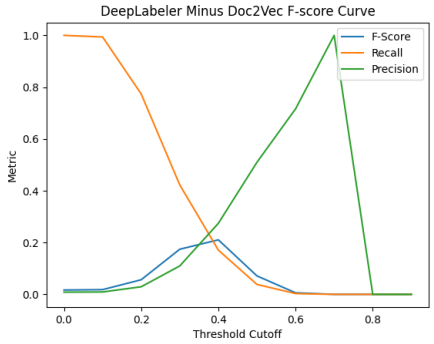

Figure 5: DeepLabeler w/o Doc2Vec Micro-F1 Score Curve

4.3 Additional results not present in the original paper

As described in the ablation, this report went out to seek if adding an embedding layer in place of the Word2Vec model would increase the micro-f1 score. This report found that by removing the Word2Vec model, the micro-f1 score decreases. See Figure 6 for the f1-scores at different thresholds. You will notice this model's precision also tanks to zero at high thresholds, similar to the second result.

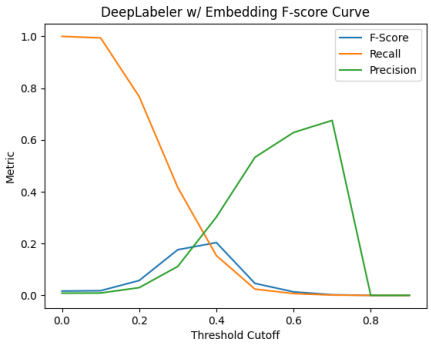

Figure 6: DeepLabeler w/ Embedding Micro-F1 Score Curve

Additionally, a DeepLabeler model was trained for 50 epochs and the training loss is reported in figure 7. The training time was roughly 2 hours and 8 seconds. The overall f1-score was improved to 0.28 at a threshold of 0.4 compared to just 0.25 f1-score with the original model.

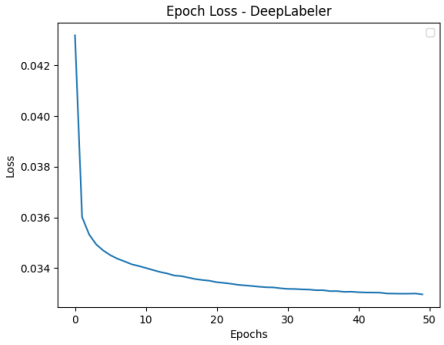

Figure 7: DeepLabeler for 50 Epochs

5 Discussion

The main result from this report do not confirm the findings of the paper. My report finds that the linear SVM model still outperforms the DeepLabeler architecture via f-score. Perhaps the large difference in the linear SVM results is the implementation of both models. I can speculate the authors did not use TF-IDF or threshold the SVM like in this report. Additionally, the implementation of the DeepLabeler architecture could have been another contributing factor. I found the architecture in the paper to be ambiguous after closer inspection and at the time of implementation. The DeepLabeler model I created could not be the same as the author's original model.

The only agreement this report found was that when removing the Doc2Vec model, the f-score

of the DeepLabeler model decreases. I agree with the authors that the Doc2Vec vectors were able to increase performance by encoding a more global context of each document.

Furthermore, I found that my ablation study did not improve the f-score of the model. I speculate that in order for me to create a better embedding layer, I would have needed a lot more data or run the model for longer epochs. I did use transfer learning to initialize the embedding weights, but I believe the Word2Vec model still did a better job of representing the embeddings due to it having more data. With the DeepLabeler, only the first 700 tokens were passed through but the Word2Vec model did not have this restriction so it was able to train on more data.

The overall implementation of DeepLabeler could also be updated with a Transformer model. Many pre-trained transformers are available and could be used to do multi-label classification. If I had more time, I would have used a pre-trained encoder model like BERT to tackle the multilabel problem (Devlin et al., 2019).

5.1 What was easy

Accessing the MIMIC-III dataset was relatively easy and painless after watching a few training videos and taking a few quizzes. Additionally, training the Word2Vec and Doc2Vec models was simple due to the implementation being open-source (Řehůřek and Sojka, 2010).

5.2 What was difficult

One major pain point of this report was interpreting the exact architecture the paper was using. It was unclear if the authors were using a 1D convolutional network or a 2D convolutional network, and what their output dimensions were after the max pool operation. Additionally, it would have been helpful to have the tokenizer process that the authors used. It was very difficult to try to reproduce the average token count provided in the paper due to not knowing what tokenizer the authors were using. It was also difficult to understand the exact setup of the linear SVM, it was unclear if the paper had used TF-IDF.

5.3 Recommendations for reproducibility

I would recommend the authors expose their code and make it open source. By making their implementation open source, it would have made the reproducibility of the report much easier by exposing their implementation of the model architecture, linear SVM, and tokenizer. As discussed before, the major pain points were having an ambiguous architecture description, no description of the linear SVM setup, and no tokenizer shared.

6 Communication with original authors

An email was sent to the author, although I received no response back. Therefore I was not able to reference their code and implementation of the paper. This report will be sent to them.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Pollard T. Mark R. Johnson, A. 2022. Mimic-iii clinical database carevue subset. *PhysioNet. https://doi.org/10.13026/8a4q-w170*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents.

Min Li, Zhihui Fei, Min Zeng, Fang-Xiang Wu, Yaohang Li, Yi Pan, and Jianxin Wang. 2019. Automated icd-9 coding via a deep learning approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(4):1193–1202.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.