ML Challenge Report
Multi-label Classification of EHRs with ICD10 Codes

Sonalika Singh
MA23C044

This report presents the solution for the DA5401 end-semester ML challenge, focusing on the automatic assignment of ICD10 codes to electronic health records (EHRs) using a multi-label classification approach. I leveraged LLM embeddings for feature representation and implemented a deep learning model optimized for multi-label classification. This document details the data engineering, model architecture, training process, and performance metrics, culminating in a test set evaluation with a Micro F2 score of 0.8183.

1 Introduction

Medical coding is the process of assigning standardized ICD10 codes to diagnose conditions in EHRs. This report documents the solution implemented for the DA5401 challenge, which involved predicting ICD10 codes from embeddings generated from outpatient (OP) medical charts. Given the multi-label nature of the data, we employed a deep learning model to handle the complexity of multiple diagnoses per record.

2 Data Engineering

2.1 Dataset Description

The dataset comprises approximately 200,000 training samples with 1024-dimensional embeddings and associated ICD10 codes. Each test sample may have multiple labels, making it suitable for a multi-label classification approach.

2.2 Data Loading and Preprocessing

- **Embedding Loading**: Loaded two sets of embeddings from `embeddings_1.npy` and `embeddings_2.npy`, combining them using `np.vstack()`.
- **Multi-hot Encoding of Labels**: ICD10 codes from the text files were parsed and converted to a multi-hot encoded format. A dictionary was created to map each unique code to an index, and labels were then converted into multi-hot vectors.
- **Train-Validation Split**: The dataset was split into 80% training and 20% validation using `train_test_split()` with a fixed random state for reproducibility.

3 Exploratory Data Analysis (EDA)

3.1 Label Distribution

A distribution of the ICD10 codes was generated to analyze label frequency, revealing an imbalanced dataset with a few codes occurring frequently and many being rare. This influenced our decision to use a threshold-based approach for final predictions.

4 Model Selection

4.1 Model Architecture

Firstly, I used a deep neural network model to manage the high-dimensional input and multi-label output, comprising:

- **Dense Layers**: Layers of sizes 2048, 1024, 512, and 256, each followed by `LeakyReLU` activations for stable gradient flow.
- **Dropout Layers**: Dropout was added at rates 0.4 and 0.3 to reduce overfitting.
- **Output Layer**: The output layer has a `sigmoid` activation function to handle the multi-label nature.

The model architecture allowed for deep feature extraction and efficient multi-label classification.

4.2 Compilation and Learning Rate Schedule

The model was compiled with:

- **Loss Function**: Binary cross-entropy, suited for multi-label classification.
- **Optimizer**: Adam optimizer with a `CosineDecay` learning rate schedule, starting at 0.0005 and reducing gradually.

5 Training and Evaluation

5.1 Training Process

The model was trained with the following parameters:

- **Epochs**: 50 epochs, increased from initial attempts, allowed for extended learning.
- **Batch Size**: Set to 128.
- **Validation Metrics**: The validation dataset was monitored for accuracy, precision, and recall.

5.2 Performance Results on Validation Set

The final model achieved:

- **Micro F2 Score on Validation Set**: 0.8183.
- **Precision and Recall**: Average precision of 0.8341 and recall of 0.8079 on the validation set.

Table 1 provides additional details from key epochs during training.

Epoch	Training Accuracy	Training Loss	Validation Precision	Validation Recall
1	0.6363	7.3614e-04	0.8341	0.8079
5	0.6373	7.2788e-04	0.8342	0.8076
10	0.6375	7.2162e-04	0.8328	0.8091

Table 1: Performance metrics during key epochs.

6 Test Set Predictions and Submission

6.1 Prediction Thresholding

For final predictions, I experimented with various threshold values (0.42, 0.45, 0.47, and 0.49) to convert probabilities into binary labels. I observed that as the threshold increased, the leaderboard scores improved in a consistent manner. This testing helped identify 0.49 as the most effective threshold for maximizing the Micro F2 score on the leaderboard, demonstrating the importance of fine-tuning thresholds in multi-label classification.

6.2 Generating Submission File

The output file for Kaggle submission was formatted as shown in Table 2. For each test embedding, the predictions were sorted alphabetically and combined into a single string separated by semicolons (;).

ID	Labels
1	G56.21
2	M65.9;S83.242A
3	G56.01
4	M65.312
5	S83.241A;S83.281A

Table 2: Example format of the submission file.

The submission file, named `submission.csv`, was generated successfully, with 99,490 rows, covering all test embeddings.

7 Alternative Model: Logistic Regression with OneVsRest

I first designed a deep neural network model with multiple dense layers and dropout layers to handle the high-dimensional embeddings and multi-label classification. This model is described in Section 4.1.

To explore a simpler baseline, I also implemented a Logistic Regression model with the OneVsRest strategy. Using this approach, each ICD10 code is treated as an independent binary classification task. Logistic Regression is generally faster to train, so it was valuable as a point of comparison.

The model was trained on the same preprocessed dataset. However, given the complexity of multi-label classification in this context, the OneVsRest Logistic Regression approach yielded lower scores, indicating it may not effectively capture complex relationships in the data compared to the deep learning model.

The Logistic Regression model achieved a micro-average F1 score of 0.259 on the Kaggle leaderboard, significantly lower than the deep learning model. The classification report for this model is shown in Table 3.

8 Training and Evaluation

8.1 Training Process for Logistic Regression Model

For the Logistic Regression model with OneVsRest, the training process involved converting probabilities to binary predictions with a custom sigmoid function. Predictions were adjusted based on a threshold of 0.5. After training, the model's performance was evaluated on the validation set.

8.2 Performance Results on Validation Set

The Logistic Regression model performed substantially lower compared to the deep learning model, achieving only a 0.259 Micro F2 score on the Kaggle leaderboard. Table 3 provides the classification report on the validation set.

Class	**Precision**	**Recall**	**F1-Score**
Class 0	0.99	0.85	0.91
Class 1	1.00	0.05	0.09
Class 2	0.00	0.00	0.00
Class 3	0.00	0.00	0.00
⋮	⋮	⋮	⋮
Class 10	0.50	0.11	0.18
⋮	⋮	⋮	⋮
Micro Avg	0.87	0.65	0.74
Macro Avg	0.48	0.30	0.34
Weighted Avg	0.81	0.65	0.70
Samples Avg	0.77	0.70	0.71

Table 3: Classification report for Logistic Regression with OneVsRest on the validation set.

8.3 Comparison of Models

While the OneVsRest Logistic Regression model offered a simpler and faster alternative, its performance was significantly lower than that of the deep learning model. This difference illustrates the importance of complex architectures for capturing relationships in multi-label classification tasks with high-dimensional embeddings

9 Conclusion

This project explored two different modeling approaches for multi-label classification of ICD10 codes from EHR data: a deep learning model with a multi-layer dense architecture and a simpler Logistic Regression model with the OneVsRest strategy. The deep learning model achieved a strong Micro F2 score of 0.8183 on the validation set, demonstrating its capability to capture complex relationships in high-dimensional medical data. In contrast, the Logistic Regression model achieved a significantly lower score of 0.259 on the leaderboard, highlighting the limitations of simpler models in handling the intricate dependencies present in multi-label tasks.

The results suggest that while Logistic Regression can provide a baseline, complex architectures such as deep neural networks are more effective for high-dimensional and multi-label datasets. Future work could include experimenting with advanced architectures like transformers, incorporating techniques to handle data imbalance, and fine-tuning hyperparameters further to optimize performance.

Overall, this project underscored the importance of model selection and highlighted the advantages of deep learning for complex, multi-label classification tasks in healthcare.

10 Summary and Learning Outcomes

Working on this project gave me hands-on experience with multi-label classification, especially using deep learning techniques for complex, high-dimensional medical data. Key takeaways from this challenge include:

- **Data Preprocessing and Encoding**: I learned how to manage and preprocess large-scale datasets and how to effectively apply multi-hot encoding for multi-label classification tasks.
- **Model Architecture Design**: Building a deep neural network model with multiple layers and dropout required careful tuning. I gained insights into balancing model depth with overfitting concerns and learned how activation functions like LeakyReLU can improve model performance.
- **Evaluation Metrics**: The use of metrics like Micro F2 score helped me understand the importance of metric selection in multi-label tasks. I also learned about the significance of precision and recall in imbalanced datasets.
- **Thresholding Techniques**: Applying probability thresholds to generate final label predictions taught me how small changes in threshold values can impact model outcomes, especially in multi-label classification.
- **Improving Model Performance**: The iterative process of training, adjusting hyperparameters, and validating on the test set helped me appreciate the trial-and-error approach needed for optimal results.

Overall, this project strengthened my skills in machine learning, data handling, and deep learning, and it provided valuable insights into the practical challenges of multi-label classification in a real-world healthcare context.

.