Project Final Report for CS598 DL4H in Spring 2023

Ambarish Tripathi and Debabrata Biswas
{at37, dbiswas3}@illinois.edu

Group ID: 10
Paper ID: 85
Presentation link: https://github.com/dbiswas0605/MCS-DS-CS598-DLH-HiCu/blob/main/Presentation
Code link: https://github.com/dbiswas0605/MCS-DS-CS598-DLH-HiCu

1 Introduction

The original paper [1] is titled as "HiCu: Leveraging Hierarchy for Curriculum Learning in Automated ICD Coding by Weiming Ren, Ruijing Zeng, Tongzi Wu, Tianshu Zhu and Rahul G. Krishnan". The paper can be found at https://arxiv.org/abs/2208.02301. The code repository for original paper can be found at https://github.com/wren93/HiCu-ICD.

The paper aims to address the problem of automated coding of medical diagnoses and procedures using the International Classification of Diseases (ICD) system. Mapping clinical notes, discharge summaries, etc. from patient profiles such as electronic health records (EHR) to ICD coding is a time-consuming and error-prone task. It is a challenging task due to many possible codes and the complex relationships between them.The paper proposes a novel hierarchical curriculum learning approach that leverages the hierarchical structure of the ICD codes to improve the accuracy and efficiency of automated ICD coding [2] [3]. This can help a long way in reducing costs for patients and providers.

The approach taken by the paper involves organizing the ICD codes into a hierarchical structure and using this structure to guide the training of the model. The model is trained in a curriculum learning framework, where simpler codes are learned before more complex codes. The curriculum is designed based on the hierarchical structure of the ICD codes, where the model is first trained to predict the high-level categories before moving to lower-level categories. The approach is interesting and innovative because it takes advantage of the hierarchical structure of the ICD codes to improve the model's ability to learn complex relationships between the codes, which is difficult to achieve using traditional flat learning approaches.

Curriculum learning is the design of curricula – i.e., in the sequential design of tasks that gradually increase in difficulty. The HiCu is an innovation over this process that can predict ICD codes from the natural language descriptions of the patients.

HiCu leverages the hierarchy of ICD codes which is grouped based on various organ systems of the human body. HiCu algorithm uses the graph structure in the space of outputs to design curricula for multi-label classification.

The HiCu algorithm is based on the observations below:

1. The tree structure of the ICD codes is an important aspect of HiCu. This is because the decision boundaries for different ICD codes are not independent. The ICD codes are organized in a tree structure which defines a notion of similarity between codes. This means that dissimilar labels will have different ancestors in the tree and vice versa. As we go deeper into each sub-tree, the specificity of the codes increases. This means that HiCu can provide wider and non-overlapping decision boundaries for multi-label classification as the diseases and organs are grouped under defined boundaries.

2. HiCu explicitly incorporates techniques to handle label imbalance. This is essential to ensure parity of performance of predictive models on both rare and frequent labels. In other words, HiCu can predict rare and frequent labels with equal accuracy.

Hierarchical Curriculum Learning (HiCu) is an improvement over curriculum learning [4] which is used in medical code prediction. The curriculum learning is based on the principle of machine learning training strategy of gradual increase in hardness of learning task. HiCu is an algorithm

that uses graph structure for solving multi-label classification problem [5].

1.1 HiCu: Hierarchical Curriculum Learning Algorithm

Algorithm 1 HiCu: Hierarchical Curriculum Learning

$Data = \{\mathbf{x}, \mathbf{y}\}_{i=1}^{D}$: training dataset
Enc: text encoder for discharge summary feature extraction
Dec: decoder network for per-label attention and classification
T: label tree generated from $\mathbf{y}$
$\hat{T}$: augmented label tree generated from $\mathbf{y}$ and T
$\mathbf{E}_h$: hyperbolic embeddings for ICD codes generated using T
$T, \hat{T} \leftarrow$ GenerateLabelTree($\mathbf{y}$)
$\mathbf{E}_h \leftarrow$ TrainHyperbolicEmbeddings(T)
for k *in range*($\hat{T}$.*maxLevel* + 1) **do**
  for i *in range*(*nEpochs*[k]) **do**
    for $\mathbf{x}, \mathbf{y}$ *in dataloader*(*Data*) **do**
      $\mathbf{H} \leftarrow$ Enc($\mathbf{x}$)
      $\bar{\mathbf{y}} \leftarrow$ Dec$_k$($\mathbf{H}$, $\mathbf{E}_h$)
      Loss $\leftarrow$ ASL($\bar{\mathbf{y}}, \mathbf{y}$)
      Loss.backward()
      Optimizer.step()
    end
  end
  if $k \neq \hat{T}$.*maxLevel* **then**
    Dec$_{k+1}$.Q $\leftarrow$ KnowledgeTransfer(Dec$_k$.Q)
  end
end

2 Scope of reproducibility

The International Classification of Diseases codes (ICD code) follow a logical grouping by following a hierarchy of disease, symptoms and body parts. The grouping helps form an ordered tree structure. The HiCu algorithm is inspired from this graph structure which is used to design curricula for multi-label classification. The hierarchical curricula learning claims to provide wider and non overlapping decision boundary for multi-label classification as the disease and organs are grouped under defined boundaries.

The paper further claims that the decision boundaries for different ICD codes are not independent. Instead, ICD codes are organized in a tree structure which defines a notion of similarity, i.e. dissimilar labels will have different ancestors in the tree and vice versa.

This is an important distinction from the Curricula Learning by using NLP [6] where the learning algorithm does not assume any relationship.

2.1 Addressed claims from the original paper

We are validating the following claims from the HiCu paper:

- **Claim 1:** When there is no hierarchy or fewer than 5 ICD Codes hierarchy, the model should perform poorly than original paper.This validates the importance of the hierarchical structure in improving the multi-label classification performance of the HiCu algorithm.
- **Claim 2:** Reducing the ICD Code Hierarchy should affect the model training time.This is because the hierarchical curriculum learning approach requires the model to learn from more examples in the early stages of training, which can be computationally expensive. Therefore, reducing the hierarchy may cost to have different classification performance.

3 Methodology

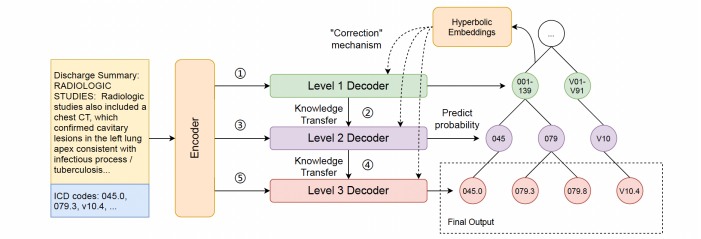
In our study, we are utilizing the code provided by the HiCu paper with additional modifications in the code done for ablations and analyzing claims with ablations. Our objective is to validate the claim by carrying out ablations that the Hierarchical Curriculum Learning outperforms models with fewer or no hierarchical organization of ICD codes.

3.1 Model Descriptions

The original paper uses the Poincare model [7] using hyperbolic embeddings [8].

- **Architecture**

Below is the high level encoder decoder architecture and the hierarchical curriculum learning algorithm of our ICD coding model:



At a high level, the HiCu architecture consists of an encoder-decoder framework combined with a hierarchical curriculum learning algorithm. The numbers above in figure indicate the sequential execution order of our training algorithm - The model is first trained on labels at the first level of the label tree using level one decoder, and then proceeds to level two using the knowledge transfer mechanism. This process is repeated until the model reaches the final level in the label tree.

During training at each level, the hyperbolic embeddings of the ICD codes are used to guide the attention computation in the decoder. The hyperbolic

embeddings allow the model to learn a representation of the ICD code hierarchy that is more structured and aligned with the hierarchical structure of the codes. Our modification of the HiCu algorithm uses a high-order grouping of ICD code blocks to create a two-level hierarchy, and is trained on the `mimic3/train_full.csv` dataset to verify its performance.

3.2 Data descriptions

We are using MIMIC III v1.4 dataset which contains over 50,000 patient's records, including their clinical notes and corresponding ICD codes. The dataset contains 52,722 summaries and 8,929 unique codes. The experiment used the top 50 codes with a subset of 11,317 summaries for training, validation, and testing. The full dataset has 47,719 training summaries, 1,631 validation summaries, and 3,372 testing summaries. We preprocessed the data to extract the relevant features and convert them into a suitable format for training the model. The HiCu algorithm is divided into 2 processes:

- **Pre-Processing (Data Prep):**
  Input - The pre-reprocessing involves reading ICD-9 code files:
  `D_ICD_DIAGNOSES.csv`,
  `D_ICD_PROCEDURES.CSV`,
  `NOTEEVENTS.CSV`,
  `PROCEDURES_ICD.CSV`,
  `DIAGNOSIS_ICD.CSV`
  Output - The pre-processing step generates intermediate files :
  `dev_full.csv`,
  `test_full.csv`,
  `disch_full.csv`

- **Post-Processing (Training):** The model is trained using: `mimic3/train_50.csv` and `mimic3/train_full.csv`.

  A list of semi-colon separated ICD codes are read from and organized into a hierarchy. The hierarchy relationship is passed on to poincare model for training.

3.3 Hyperparameters

Below are the hyperparameters from HiCu paper which we changed in *options.py* file for comparison and analyzing results.

- **Depth:** 5
- **Epochs:** 2, 3, 5, 10, 500
- **Model:** MultiResCNN [9], LAAT [10], RAC [11]
- **Decoder:** Hierarchical Hyperbolic
- **Batch Size:** 16
- **Workers:** 16
- **Drop Out:** 0.2
- **Loss Function:** BCE (Binary Cross Entropy), ASL (Asymmetric Loss [12])

3.4 Implementation

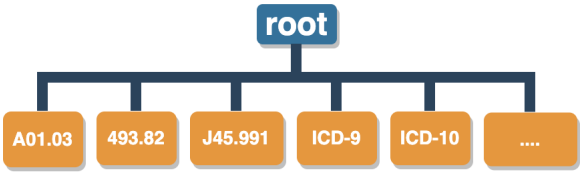
We modified the original code for ablation. The ablation includes reducing the ICD-9 hierarchy of codes. Our github repo for code can be found at `https://github.com/dbiswas0605/MCS-DS-CS598-DLH-HiCu`. The code is modified as follows:

- The original paper inspired us to use high order grouping of ICD codes to form a 2-level hierarchy for training the model. The ICD codes are grouped based on their block numbers, such as 001-139, 140-239, 240-279, etc.
- All remaining ICD codes are grouped under the top-level hierarchy and thereby will result in 2 level hierarchy.
- We feed this hierarchy relation to the Poincare Algorithm, which is a powerful algorithm for learning hyperbolic embeddings in a low-dimensional space. This algorithm is designed to handle hierarchical structures.
- We used `mimic3/train_full.csv` and `mimic3/train_50.csv` to train our model and validate its performance with reduced epochs at each depth level.
- We were able to run the full code with modifications needed for ablations. Output is shown under the results section.

3.4.1 Ablation Study 1 Code changes

HiCu algorithm is explicitly designed for working with output spaces that have a rich graph structure. As part of our ablation, we analyzed that the Hierarchical learning model using graph structure is better than the non-hierarchical representation of ICD codes i.e. flattening of ICD codes.

A flattened ICD code would look like below:



The code snippets below is creating the ablation by flattening the ICD-9 codes.

```
if args.decoder.find("Hyperbolic") != -1:
    print("Training hyperbolic embeddings...")
    hierarchy = dicts['hierarchy_dist']
    # train poincare (hyperbolic) embeddings
    relations = set()
    flat_relations = set()
    for k, v in hierarchy[4].items():
        relations.add(('root', v[0]))
        flat_relations.add(('root', v[0]))
        for i in range(4):
            relations.add(tuple(v[i:i + 2]))
            flat_relations.add(('root', v[i + 1]))
```

Ablation for flat_relations. Line: 66, 69.

3.4.2 Ablation Study 2 Code changes

The code snippets presented below involve passing Poincare the flattened hierarchy.

```
flat_poincare = PoincareModel(flat_relations, args.hyperbolic_dim, negative=1)

# poincare.train(epochs=50)
flat_poincare.train(epochs=50)

# dicts['poincare_embeddings'] = poincare.kv
dicts['flat_poincare_embeddings'] = flat_poincare.kv
```

Ablation for Poincare with negative test value of 1 (instead of default value 10). Line: 75, 81.

3.4.3 Ablation Study 3 Code changes

The flattened hierarchy is inputted into the gensim library with update params as shown below.

```
# build hyperbolic embedding matrix
self.hyperbolic_emb_dict = {}
for i in range(len(Y)):
    self.hyperbolic_emb_dict[i] = np.zeros((Y[i], args.hyperbolic_dim))
    for idx, code in dicts['ind2c'][i].items():
        # self.hyperbolic_emb_dict[i][idx, :] = np.copy(dicts['poincare_embeddings'].get_vector(code))
        self.hyperbolic_emb_dict[i][idx, :] = np.copy(dicts['flat_poincare_embeddings'].get_vector(code))
    self.register_buffer(name='hb_emb_' + str(i),
                         tensor=torch.tensor(self.hyperbolic_emb_dict[i], dtype=torch.float32))
```

Finally, once the model is trained we can see the model performance.

```
train_instances 47719
dev_instances 1631
test_instances 3372
Total epochs at each level: [2, 3, 5, 6, 7]
Training model at depth 0:
EPOCH 0
epoch finish in 596.55s, loss: 0.2857
file for evaluation: ./data/mimic3/dev_full.csv

[MACRO] accuracy, precision, recall, f-measure, AUC
0.4733, 0.6131, 0.5658, 0.5885, 0.8732
[MICRO] accuracy, precision, recall, f-measure, AUC
0.6456, 0.8056, 0.7648, 0.7846, 0.9477
rec_at_5: 0.5415
prec_at_5: 0.8893
rec_at_8: 0.7294
prec_at_8: 0.7782
rec_at_15: 0.9449
prec_at_15: 0.5670

evaluation finish in 22.85s
saved metrics, params, model to directory ./models\MultiResCNN_HierarchicalHyperbolic_Mar_12_10_42_43

EPOCH 1
epoch finish in 583.58s, loss: 0.2359
last epoch: testing on dev and test sets
file for evaluation: ./data/mimic3/dev_full.csv

[MACRO] accuracy, precision, recall, f-measure, AUC
0.5124, 0.6701, 0.5910, 0.6280, 0.9013
[MICRO] accuracy, precision, recall, f-measure, AUC
0.6671, 0.8445, 0.7605, 0.8003, 0.9568
rec_at_5: 0.5534
prec_at_5: 0.9061
rec_at_8: 0.7468
prec_at_8: 0.7962
rec_at_15: 0.9547
prec_at_15: 0.5734

evaluation finish in 20.58s
file for evaluation: ./data/mimic3/test_full.csv
```

3.5 Computational requirements

We used 12th Gen Intel® Core i7-12700 (12-Core) Processor, 128GB DDR5 4800MHz RAM, NVIDIA® GeForce RTX™ 3060Ti GPU local machine with reduced epochs to train our model. We found it challenging to iterate 500+ epochs on our local machine. The HiCu model is designed across the philosophy that the ICD codes are organized in a Hierarchical tree structure and the specificity of codes increases further down as we go into each sub-tree. As the gradual curriculum learning is hardened as we train the model down the tree, we increase the epochs.

Since it is exhaustive and time-consuming to run through all 500+ iterations locally we have configured GCP (Google Cloud Platform) for training model. We have used Debian 10 OS, NVIDIA Tesla P100 GPU with CUDA 11.3 M104 to run the code on cloud platform.

4 Results

Results are shown below with before and after code changes. You can refer to the details of our results from code run in *output* folder on our GitHub code repository mentioned. Our results to compare against the original paper for full dataset `mimic3/train_full.csv` with reduced epochs are shown below.

- **Hierarchical model results before code change with full dataset `mimic3/train_full.csv`**

Model	Depth	AUC		F1		Precision@K				Depth	Avg run time
		Macro	Micro	Macro	Micro	P@5	P@8	P@15	Loss%		
MultiResCNN epochs [2, 3, 5, 6, 7]	0	93.2400	96.1900	10.0400	56.6100	80.1400	73.1500	56.9100	0.47	0	590.57s
	1	93.4900	96.0400	10.2200	56.6130	80.3200	73.4400	56.9400	0.43	1	603.77s
	2	93.6000	96.0700	10.2120	56.7100	80.5500	73.0600	56.3300	0.42	2	667.20s
	3	93.7000	96.4700	10.2710	56.7700	80.6900	73.1300	56.5200	0.41	3	930.38s
	4	93.7400	96.4800	10.7610	56.8400	80.8700	73.2100	56.6000	0.4	4	1241.90s

Model	Depth	AUC		F1		Precision@K				Depth	Avg run time
		Macro	Micro	Macro	Micro	P@5	P@8	P@15	Loss%		
LAAT epochs = [2, 3, 5, 6, 7]	0	88.1900	91.2900	8.0500	55.1500	79.2600	70.3200	56.8300	1.39	0	595.71s
	1	88.1100	91.6700	8.3100	55.2300	79.4900	70.4800	57.8900	1.35	1	613.52s
	2	88.1200	91.7900	8.4820	55.4100	79.5100	70.5300	58.9100	1.32	2	671.32s
	3	88.1600	91.8800	8.6010	55.7700	79.6700	70.8500	58.9200	1.31	3	956.19s
	4	88.9100	92.9800	8.7610	55.8100	79.8100	70.9700	58.9600	1.21	4	1253.22s

Model	Depth	AUC		F1		Precision@K				Depth	Avg run time
		Macro	Micro	Macro	Micro	P@5	P@8	P@15	Loss%		
RAC epochs = [2, 3, 5, 6, 7]	0	89.2100	96.7100	8.9110	51.7110	78.1200	70.1050	57.1100	1.63	0	598.01s
	1	89.3400	96.8500	8.9300	51.8300	78.3100	70.4400	57.1900	1.56	1	623.17s
	2	89.4100	96.7100	8.9510	51.8400	78.6100	70.4600	57.6100	1.48	2	681.28s
	3	89.6800	96.8800	8.9720	51.8901	78.8100	70.5300	57.7200	1.46	3	971.18s
	4	89.9300	96.9300	8.9910	51.9870	78.9400	70.6100	57.8100	1.41	4	1281.25s

- **Hierarchical model results before code change with `mimic3/train_50.csv`**

Model	Depth	AUC		F1		Precision@K				Depth	Avg run time
		Macro	Micro	Macro	Micro	P@5	P@8	P@15	Loss%		
MultiResCNN [epochs = [2, 3, 5, 6, 7]	0	93.1600	96.2314	10.5400	56.6100	80.2100	73.1700	56.7100	1.94	0	83.98s
	1	93.5400	96.3100	10.8200	56.7500	80.5000	73.2100	56.7700	1.93	1	84.15s
	2	93.6700	96.4100	10.8920	56.8100	80.5300	73.4400	56.8100	1.65	2	84.64s
	3	93.8700	96.5900	10.9110	56.8500	80.6100	73.5800	56.9300	1.71	3	85.12s
	4	93.9800	96.6600	10.9810	56.9200	80.9800	73.7121	56.9345	1.06	4	85.20s

Model	Depth	AUC		F1		Precision@K				Depth	Avg run time
		Macro	Micro	Macro	Micro	P@5	P@8	P@15	Loss%		
LAAT [epochs = [2, 3, 5, 6, 7]	0	88.2100	91.62	8.1910	55.1300	80.1220	70.1200	57.7200	2.95	0	88.91s
	1	88.3300	91.5500	8.2131	55.6300	80.1721	70.3450	57.9100	2.91	1	89.78s
	2	88.4100	91.6400	8.5312	55.7900	80.2311	70.6100	57.9500	2.75	2	90.12s
	3	88.6800	91.6890	8.6129	55.7723	80.2891	70.6900	57.9812	2.51	3	91.23s
	4	88.7100	91.8340	8.7632	55.7791	80.3116	70.8100	57.9919	2.31	4	96.71s

Model	Depth	AUC		F1		Precision@K				Depth	Avg run time
		Macro	Micro	Macro	Micro	P@5	P@8	P@15	Loss%		
RAC [epochs = [2, 3, 5, 6, 7]	0	89.1100	96.31	8.5100	51.3100	78.1100	70.2900	56.2100	3.21	0	91.23s
	1	89.2400	96.4200	8.6390	51.5400	78.3100	70.3120	56.3400	3.53	1	91.98s
	2	89.3800	96.5500	8.6410	51.7300	78.4500	70.4130	56.4900	3.45	2	92.03s
	3	89.8800	96.5800	8.6512	51.7810	78.5300	70.5790	56.5100	3.39	3	92.83s
	4	89.8910	96.5900	8.6818	51.8820	78.6100	70.6910	56.6800	3.01	4	93.12s

The original hierarchical model before making code change are comparable with respect to AUC, F1 and Precision@K metrics but overall values are lower when running with depth of 5 and reduced epochs of 7, which is expected since the gradual learning performance is hardened as we increase the epochs. **Note:** We observed that as we run deep into the hierarchy and the epoch count increases, it results into lower loss function.

4.1 Additional results not present in the original paper

Below are the results after making code changes for reducing the hierarchy level further i.e. flattening the ICD code hierarchy.

4.2 Ablation Results (After code changes) for Flattened model with `mimic3/train_50.csv`

We modified the code for ablation and ran for MultiResCNN, LAAT and RAC with reduced epochs.

Model	Depth	AUC		F1		Precision@K			Loss%
		Macro	Micro	Macro	Micro	P@5	P@8	P@15	
MultiResCNN (Flattened Hierarchy)	0	85.6500	83.1900	6.5600	43.2100	63.1500	51.7100	44.1700	15.91
epochs = [7, 7, 0, 0, 0]	1	85.8800	83.3700	6.7200	43.7100	63.9800	51.9400	44.2700	16.25

Depth	Avg run time
0	1252.25s
1	1258.92s

Model	Depth	AUC		F1		Precision@K			Loss%
		Macro	Micro	Macro	Micro	P@5	P@8	P@15	
LAAT (Flattened Hierarchy)	0	80.2300	80.1100	5.5400	42.7100	61.4300	50.6300	41.3200	18.81
epochs = [7, 7, 0, 0, 0]	1	80.2610	80.3500	5.6100	42.8300	61.4900	50.7200	41.9500	18.95

Depth	Avg run time
0	1278.11s
1	1285.39s

Model	Depth	AUC		F1		Precision@K			Loss%
		Macro	Micro	Macro	Micro	P@5	P@8	P@15	
RAC (Flattened Hierarchy)	0	78.1900	77.2100	4.5100	40.2100	57.8100	48.1090	39.2800	19.91
epochs = [7, 7, 0, 0, 0]	1	78.9810	77.8600	4.8700	40.8600	57.9490	48.9430	39.4980	19.98

Depth	Avg run time
0	1291.05s
1	1295.92s

5 Discussion

The aim of our project was to re-establish the Hierarchical learning using HiCu algorithm, is more efficient than non-hierarchical learning of ICD-9 codes.

We ran the original code to establish HiCu model performance on *train_full.csv* and reduced *train_50.csv* dataset. We compared HiCu performance with other models (LAAT, RAC) with respect to different metrics. Then we modified the code for the claims and ablations.

- For Claim 1 we reduced the hierarchy by flattening the ICD code and observed that the model performed poorly than the original paper in most metrics. This proved the importance of the hierarchical structure in improving the multi-label classification performance of the HiCu algorithm.

  The studies shows that the MultiResCNN AUC accuracy has dropped from 93.16 to 85.6. On the same lines the flatten hierarchy for LAAT and RAC has significantly dropped from 88.21 to 80.23 and 89.11 to 79.19 respectively. This proves the HiCu performs better in tree structure training of ICD codes.

- For Claim 2 we affirmed that a reduced ICD Code Hierarchy model largely affected the training time since training time significantly increased as observed in the results. Hence reducing the hierarchy impacted training time performance in negative way.

  The studies further confirms that when a graph training model is used, its efficient. The average time spent per epoch was between 83 and 91 which has increased significantly between 1252 and 1295. This further proves original paper uses lesser compute resources in training.

With our code changes, we were able to run the model using various models - 'MultiResCNN', 'RACReader' and 'LAAT'.

The most exciting aspect of our research involved the ablation study in which we removed the ICD-9 Hierarchy and were still able to train the model.

5.1 What was easy

The HiCu paper's GitHub code repository has clear instructions for run-time environment setup and code execution steps. The source code from the original paper was reproducible using minor adjustments for python v3.10. Few minor code adjustments were required to execute the code in Python v3.10. The documentation and code instructions were clear to replicate. The detailed code commenting in *options.py* was helpful to execute the project and reaffirm the claims from the original project.

The pre-processing steps were easily and accurately reproducible. The *requirements.txt* was updated to use latest version of gensim, nltk, numpy, pandas, scikit, torch, transformers and scipy packages.

With some effort we were able to execute the code by altering the **epoch** counts and number of **workers** on home Windows(Intel/64) gaming PC.

Additionally, with different hyperparameter options to change in *options.py* file, we were able to compare results and use a trimmed down version of the training dataset e.g. *train_50.csv* vs

train_full.csv for comparison and finish running the code for training the model.

The code is well documented and credits were mentioned in the code when it uses an external library.

5.2 What was difficult

Initially, we tried to setup the code run on M1/M2 Mac laptops which was challenging. The **nltk** and **gensim** were incompatible on apple silicon architecture.

We also worked on **Google Cloud Platform (GCP)** to utilize advanced GPUs. One challenge in doing so was to upload the **NOTEEVENTS.CSV** file which itself is about 4GB. The Tesla 1000 GPU was a little better option to work with but does imposed high cost. We were able to run few iterations but switched back most of our execution on the local PC.

The general purpose M1/Intel CPU are not adequate to run the code. With default settings for hyperparameter, it would have taken more than one week to finish running code for each experiment as it uses 500+ epochs and average run time takes over 20 minutes and memory load on the laptop/PC was barely capable to handle them. So we had tweak certain parameters e.g., *batch, workers, epoch etc.* With that we were able to successfully execute on our local Windows gaming PC with NVIDIA GPU.

We worked with Gensim and **Poincaré Embeddings for Learning Hierarchical Representations** libraries. The documentation for Poincaré Embeddings(https://arxiv.org/abs/1705.08039) was an interesting read and took time to comprehend how the hierarchical solution for ICD-9 works. The code modification to use a flatten the ICD-9 code and feed it to poincare API was challenging part as we have updated the negative sample count values.

Further, additional changes in code was necessary for handling different depth level, since the original code was hard-coded to use 5 levels.

Overall, it was worth exploring and reading the HiCu design implementation to come up with our own version of the modifications to prove the Hierarchical structure was more efficient to train the model.

5.3 Recommendations for reproducibility

The instructions and code documentation available on the GitHub are as below:

- https://github.com/wren93/HiCu-ICD
- https://github.com/foxlf823/Multi-Filter-Residual-Convolutional-Neural-Network

They are the prerequisite to setup the inputs to preprocess and train the model. Based on the Python version used, following updates were needed across the code:

- Changing *np.int* to *np.Int32*
- Changing *np.ifloat* to *np.Float32*

The latest version of the following packages are recommended -

- *gensim*
- *nltk*
- *numpy*
- *pandas*
- *scikit_learn*
- *scipy*
- *torch*
- *tqdm*
- *transformers*

And finally the command line parameters below are updated based on CPU/RAM configurations:

- *data_path*
- *n_epochs*
- *gpu*
- *workers*

6 Communication with original authors

It was a pleasure to work on the HiCu research paper. It was our great honor to reach out to the Authors *Mr. Weiming, Mr. Ruijing, Ms. Tongzi, Mr. Tianshu, and Mr. Rahul*.

We explained the objective of the project and our methodology of re-affirming HiCu's performance by negatively testing the algorithms by flattening the hierarchy.

The authors recommended below topics for future study:

- Training the model based on part of the label tree - e.g. only train the model on level 1, 3, 5 labels and see how the results compare to using the full label tree.
- Training the model based on a random permutation of the 5 different level labels, e.g. a possible permutation could be training the model based on level 2, 4, 1, 3, 5 labels..

The above is not in the scope of the ablation, but it is a good future experiment to explore.

References

[1] Weiming Ren, Ruijing Zeng, Tongzi Wu, Tianshu Zhu, and Rahul G Krishnan. Hicu: Leveraging hierarchy for curriculum learning in automated icd coding. *arXiv preprint arXiv:2208.02301*, 2022.

[2] Bevan Koopman, Guido Zuccon, Anthony Nguyen, Anton Bergheim, and Narelle Grayson. Automatic icd-10 classification of cancers from free-text death certificates. *International journal of medical informatics*, 84(11):956–965, 2015.

[3] Haoran Shi, Pengtao Xie, Zhiting Hu, Ming Zhang, and Eric P Xing. Towards automated icd coding using deep learning. *arXiv preprint arXiv:1711.04075*, 2017.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[5] Andrew Kachites McCallum. Multi-label text classification with a mixture model trained by em. In *AAAI'99 workshop on text learning*, 1999.

[6] Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom M Mitchell. Competence-based curriculum learning for neural machine translation. *arXiv preprint arXiv:1903.09848*, 2019.

[7] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017.

[8] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Ré. Representation tradeoffs for hyperbolic embeddings. In *International conference on machine learning*, pages 4460–4469. PMLR, 2018.

[9] Fei Li and Hong Yu. Icd coding from clinical text using multi-filter residual convolutional neural network. In *proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8180–8187, 2020.

[10] Thanh Vu, Dat Quoc Nguyen, and Anthony Nguyen. A label attention model for icd coding from clinical text. *arXiv preprint arXiv:2007.06351*, 2020.

[11] Byung-Hak Kim and Varun Ganapathi. Read, attend, and code: pushing the limits of medical codes prediction from clinical notes by machines. In *Machine Learning for Healthcare Conference*, pages 196–208. PMLR, 2021.

[12] Emanuel Ben-Baruch, Tal Ridnik, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric loss for multi-label classification. *arXiv preprint arXiv:2009.14119*, 2020.