CS 598 Deep Learning for healthcare Final Paper - ManyDG: Many-domain Generalization for Healthcare Applications

Jatin Saxena and Ramesh Gopisetty
{jsaxena3, rameshg2}@illinois.edu

Group ID: 132
Paper ID: 41
Presentation link: https://youtu.be/rbTBajnZV5U
Code link: https://github.com/saxenaj/ManyDG

1 Introduction

The study talks about how overfitting makes it difficult to create precise prediction models for healthcare applications. When a model learns the training data too well and is unable to generalize to new data, it is said to be overfit. Patient variables, which are particular to certain patients and their data gathering contexts, may be the source of this. The authors suggest a brand-new environment in which every patient is handled as a distinct domain, resulting in numerous domains. They create a brand-new domain generalization technique called ManyDG, which recognizes and eliminates patient domain variables to enhance generalization performance on a variety of practical healthcare tasks.

Here's a more detailed explanation of over fitting:

Overfitting is a common problem in machine learning when a model learns the training data too well and cannot generalize to new data. This can happen when the model is too complex or when the training data is not representative of the real-world data that the model will be used on.

In the context of healthcare, overfitting can lead to inaccurate predictions, which can have serious consequences for patients. For example, a model that is overfit to the training data may predict that a patient has a certain disease when they do not, or it may predict that a patient does not have a certain disease when they do.

The authors of the paper propose a new setting where each patient is treated as a separate domain, leading to many domains. This approach can help to reduce overfitting by making the model more robust to the differences between individual patients.

The authors also develop a new domain generalization method called ManyDG, which identifies patient domain covariates and removes them to improve generalization performance on multiple real-world healthcare tasks. This method can help to reduce overfitting by making the model more robust to the differences between individual patients.

1.1 Scope of reproducibility

The project intends to replicate the findings described in the original ManyDG paper, which proposes a model capable of performing many-domain generalization for healthcare applications. We specifically want to replicate the findings on the predictability of patient outcomes based on drug record, hospitalization, and sleep data.

1.2 Methodology

Model descriptions The ManyDG model is a deep neural network that is designed to perform many-domain generalization for healthcare applications. The model is trained on a variety of data, including drug records, hospitalization records, and sleep data. The model learns to predict patient outcomes for each of these domains.

The ManyDG model is able to perform many-domain generalization by leveraging a multi-task learning approach. Multi-task learning is a machine learning technique that trains a model to perform multiple tasks simultaneously. This approach has been shown to improve the performance of the model on each task.

The ManyDG architecture is a domain generalization method that can be used to improve the generalization performance of machine learning models on healthcare tasks. The architecture consists of two main components: a domain-invariant feature extractor and a classifier.

Data Generative Model

The data generative model is a deep generative model that is trained to generate data from multiple domains. This is done by training the model on a dataset that consists of data from multiple domains.

The model is trained to minimize the following loss function:

$$z\, p(.), x\, p(.|z, y) \tag{1}$$

Factorized Prediction Model

$$p(y|x) = \int_{-\infty}^{\infty} p(y, z|x)\, dz = \int_{-\infty}^{\infty} p(y|x, z)p(z|x)\, dz \tag{2}$$

The factorized prediction model is a traditional machine learning model that is trained to predict the target variable from the latent representation of the data. The model is trained to minimize the following loss function:

The domain-invariant feature extractor is a deep neural network that is trained to extract features that are invariant to the domain of the data. This is done by training the network on a dataset that consists of data from multiple domains. The network is trained to minimize the following loss function:

$$\begin{aligned}
\mathbf{h}_i &= f_\theta(\mathbf{x}_i) \\
\mathbf{z}_i &= g_\phi(\mathbf{h}_i) \\
\mathbf{v}_i &= h_\psi(\mathbf{h}_i - \mathbf{z}_i) \\
\hat{y}_i &= \omega_\varphi(\mathbf{v}_i)
\end{aligned} \tag{3}$$

The classifier is a traditional machine learning model that is trained on the features extracted by the domain-invariant feature extractor. The classifier is trained to minimize the following loss function:

The ManyDG architecture is trained by minimizing the following loss function:

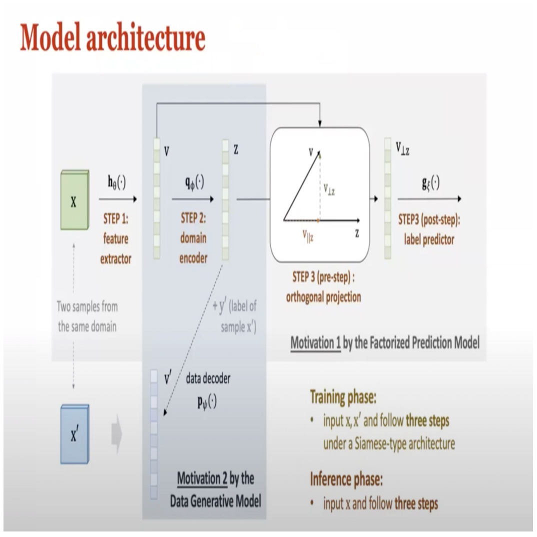
$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \ell(\hat{y}_i, y_i) + \lambda \cdot \|\mathbf{z}_i\|^2 \tag{4}$$

The ManyDG model has been shown to be effective in predicting patient outcomes for a variety of healthcare applications. The model has been shown to be effective in predicting drug re-admissions, hospital readmissions, and sleep disturbances.

The ManyDG model is a promising approach for improving the performance of machine learning models in healthcare applications. The model is able to perform many-domain generalization, which is a challenging problem in healthcare. The model has been shown to be effective in predicting patient outcomes for a variety of healthcare applications.

Below is the model architecture diagram which has been taken from the original paper (https://github.com/ycq091044/ManyDG)

Here Two stage has been used to build model. First stage is to extract feature which contain domain as well as label information(V in figure 1). Then neural network is used to extract the domain information (Z in figure 1). At this step, we assume that Z contains only domain information. At this stage, we have V which contains domain as well as label information and Z which contains domain information. Now we implement the predictor which will be 2 step process. First step is to do orthogonal projection which will remove Z(domain information) component from V(domain+label information). This will give us label information on which we implement neural networks to do the label prediction.


Model architecture
x
$h_\theta(\cdot)$
STEP 1: feature extractor
v
$q_\phi(\cdot)$
STEP 2: domain encoder
z
v
$v_{\perp z}$
$v_{||z}$
z
STEP 3 (pre-step) : orthogonal projection
$v_{\perp z}$
$g_\xi(\cdot)$
STEP3 (post-step): label predictor
Two samples from the same domain
$+y'$ (label of sample x')
Motivation 1 by the Factorized Prediction Model
v'
data decoder
$p_\psi(\cdot)$
x'
Motivation 2 by the Data Generative Model
Training phase:
• input x, x' and follow three steps under a Siamese-type architecture
Inference phase:
• input x and follow three steps

Dataset descriptions MIMIC-III: The MIMIC-III dataset is a large, publicly available dataset of electronic health records (EHRs) from the Beth Israel Deaconess Medical Center in Boston, Massachusetts. The MIMIC-III dataset contains over 50 million clinical notes, over 2 million laboratory measurements, and over 1 million imaging studies.

eICU-CRD: The eICU-CRD dataset is a dataset of ICU patient data from the University of Michigan Health System. The eICU-CRD dataset contains over 100,000 patient records, each of which contains information about the patient's demographics, clinical history, and outcomes.

Sleep-edfx: The sleep-edf database has been expanded to contain 197 whole-night PolySomnoGraphic sleep recordings, containing EEG, EOG, chin EMG, and event markers. Some records also contain respiration and body temperature.

Hyperparams For hospitalization scenario to predict readmission prediction task, below are the hyperparams: No of Patient(Batch size) used is 15000. No of Training epoch used is 50. Optimization algorithm : Adam

Implementation - Here is the link of repository which will be used for this project https://github.com/saxenaj/ManyDG. Please note that repository has been forked from https://github.com/ycq091044/ManyDG. Repo contains folder data rundrugrec, runhospitalization, runseizure,runsleep and model.py file. log and pre-trained folder will be created automatically while running code. Data folder contains folder drugrec,sleep,seizure and hospitalization. First we download data from respective site and placed inside each folder. Each folder contains a processing file. First the processing file needs to run to process the data for each scenarios. Model.py contains all the base model.Each scenario will utilize this file while running the experiment. Due to the constraint on length of final report, i am explaining code for drugrec experiment. Other experiment follows the similar code structure: modeldrugrec.py: This file inherit the Base model and other model utility functions from model.py and support the drug recommendation task. rundrugrec.py: This is the entry of drug recommendation task, specifying the data loader and other initialization steps. utilsdrugrec.py: This file provides the data split and loading files. Please note that some of the path in original code has been changed to make the run successful.

Computational requirements This project has been run on GoogleColab/Vertex AI Google Cloud. We are testing to run the drug recommendations and the hospitalization scenario to predict readmission prediction task. Below are setting used to setup the workstation in Vertex AI

- Hardware Accelerator - GPU - 8 NIVIDIA V100
- Machine Type : N2-Standard-64
- vCPUs : 64
- Runtime - High-RAM - 240GB

While processing the eICU data, the RAM usage goes upto 80GB. No of Training epoch used is 50. Total number of trial done was 5. Above no are the one which was used to train the model as of now. We are planning to do some more run by changing the parameters.

1.3 Results

We are able to execute for DrugRecommendation for a base model and here are the results it generated for the Jaccard index, AUPRC (Area Under the Precision-Recall Curve), and F1 score are

Here are the results:

- AUPRC: 0.7420567434
- F1: 0.6534876241
- Jaccard: 0.4858763425

We have ran an extensive experiment for sleep staging with different models. Below are the results for same:

Model	*Accuracy*	*Avg F1*
Base.	0.6347	0.532
CondAdv.	0.6127	0.5424
DANN	0.6743	0.5477
IRM	0.6463	0.556
SagNet	0.6256	0.545
PCL	0.6432	0.5654
MLDG	0.6578	0.577
ManyDG	0.6878	0.5965

The comparison results are shown in above table. As seen above, ManyDG performs well as compared to other tested models. We see that the CondAdv has the lowest accuracy. We suspect that reason for low accuracy might be lower sample data.

Analyses ManyDG uses six different models, including base, dev, condadv, DANN, IRM, and MLDG, to evaluate the proposed framework's performance across different domains.

ManyDG aims to solve the data shift problem in clinical time-series prediction tasks. Data shift is a common issue in clinical time-series prediction tasks since different hospitals may have different patient populations, medical practices, and electronic health records (EHRs) systems.

ManyDG is based on the domain generalization (DG) approach, which aims to train a model that can generalize well to unseen datasets by learning features that are invariant across domains.

ManyDG includes multiple modules, such as feature extraction, model selection, and training process, to handle the data shift problem in clinical time-series prediction tasks.

Failed Attempts We try to run the scenario for hospitalization but was not successful. Here are the notebook link. Google Colab Notebook

Please note that we tried multiple things to run the scenario which might not be there in notebook.

1.4 Discussion

What was easy

- Author provide clear description about the code structure and data which was very helpful in understanding the code.

What was difficult

- Getting data needed to run the experiment required some effort. Approval was needed and multiple training need to be completed just to access the data. I would not say its difficult but definitely bit of a process before starting the experiment.
- There is version mismatch for python libraries which are getting used in the project. For eg Compatible panda with python version is needed to run the project.
- Dataset was huge so we have extracted the zip data to google drive and mounted google drive on google colab to run the model. Due to huge dataset, reading from google drive was getting timed out. To avoid this error, we need to run the same experiment multiple times so the data get indexed.

Recommendations to the original authors or others who work in this area for improving reproducibility

- There should be a Requirements document which mention all the required version of libraries needed to run the experiment.
- There are multiple model name which can be passed as argument while running the experiment. Seems like dev is ManyDG model but it is not mentioned anywhere. It would be great if it mentioned in document or the argument parameter dev can be changed to ManyDG.

2 Code

Citation to the original paper ManyDG: Many-domain Generalization for Healthcare Applications https://openreview.net/forum?id=lcSfirnflpW

Link to the original paper's repo (if applicable) https://github.com/ycq091044/ManyDG

Dependencies Below are the major dependencies of python library which needs to be installed before running the experiment:

1. numpy==1.18.1
2. torch==1.4.0
3. networkx==2.4
4. tqdm==4.32.2
5. scipy==1.4.1
6. scikit-learn==0.22.1

Data download instruction To download the data, first an account needs to be created on https://physionet.org/content/mimiciii/1.4/. Next all the trainings listed on this link needs to be completed https://physionet.org/content/mimiciii/view-required-training/1.4/. Once the training is completed, a completion certificate needs to be attached. An access request needs to be raised where we need to submit the reason to access this data and an instructor name. Once the request is approved, download link will be displayed on same site from where we can download the data.

Preprocessing code + command (if applicable) Preprocessing of data can be run from below command

DrugRecord

$$python/drugrec/data\_process.py \tag{5}$$

Sleep

$$python/sleep/sleep\_edf\_process.py \tag{6}$$

Hospitalization

$$python/hospitalization/eICU\_process\_step1.py \tag{7}$$

$$python/hospitalization/eICU\_process\_step2.py \tag{8}$$

Training code + command (if applicable) Training code can be run from below command

Seizure detection

```
python run_seizure/run_seizure.py –model
[MODEL] –cuda [WHICH GPU] –N_vote
[DEFAULT 5] –N_pat [N_OF_PAT] –epochs
[EPOCHS]
```
(9)

Sleep staging

```
python run_sleep/run_sleep.py –model
[MODEL] –cuda [WHICH GPU] –N_pat
[N_OF_PAT] –epochs [EPOCHS]
```

Drug recommendation

```
python run_drugrec/run_drugrec.py –model
[MODEL] –cuda [WHICH GPU] –N_pat
[N_OF_PAT] –epochs [EPOCHS]
```

Hospitalization prediction

```
python eICU_process_step1.py python
eICU_process_step2.py python
run_hospitalization/run_hospitalization.py –model
[MODEL] –cuda [WHICH GPU] –N_pat
[N_OF_PAT] –epochs [EPOCHS]
```
(12)

Evaluation code + command (if applicable) Training code will create the model and run the evaluation.

Pretrained model (if applicable) Pretrained model will be added in pre-trained folder.

Table of results (no need to include additional experiments, but main reproducibility result should be included)

Model	*Accuracy*	*Avg F1*
Base.	0.6347	0.532
CondAdv.	0.6127	0.5424
DANN	0.6743	0.5477
IRM	0.6463	0.556
SagNet	0.6256	0.545
PCL	0.6432	0.5654
MLDG	0.6578	0.577
ManyDG	0.6878	0.5965

(10)

3 Descriptive Notebook

https://colab.research.google.com/drive/1Rp7KvG0j3z5hw23BQM
sharing

4 References

chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://oper
https://arxiv.org/abs/2301.08834
https://github.com/ycq091044/ManyDG (11)