Title
Opportunistic Learning: Algorithms and Methods for Cost-Sensitive and Context-Aware Learning

Author
Kachuee, Mohammad

Publication Date
2020

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Opportunistic Learning:
Algorithms and Methods for Cost-Sensitive and Context-Aware Learning

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Mohammad Kachuee

2020

ABSTRACT OF THE DISSERTATION

Opportunistic Learning:
Algorithms and Methods for Cost-Sensitive and Context-Aware Learning

by

Mohammad Kachuee
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2020
Professor Majid Sarrafzadeh, Chair

Classical approaches to machine learning sought to improve the efficiency and accuracy of prediction but often failed to account for the costs associated with the collection of data and expert labels. This shortcoming is particularly limiting in the smart health setting, where accurate classification often requires an invasive level of information querying. Furthermore, in domains such as medical diagnosis, appropriate data should be collected based on a scientific hypothesis, and ground-truth labels may only be provided by highly trained domain experts. Additionally, in many studies, informative features are not scientifically predetermined, and usually, there are many information sources that can be considered as hypothetical relevant features that including all of them is not practical.

In order to address these issues, we suggest novel end-to-end solutions considering different aspects of a real-world learning system. Specifically, we consider feature acquisition, labeling, model training, and prediction at test-time as different aspects of a system that tries to achieve the goal of making accurate predictions efficiently. In this paradigm, information is acquired incrementally based on the value it provides and the cost that should be paid for acquiring it. In this thesis, we explore dynamic and context-aware information acquisition techniques to collect the right piece of information at the right time. Additionally, as inference using incomplete data is an inevitable part of such methods, we propose a novel approach to not only impute missing values but also to capture prediction uncertainties.

The dissertation of Mohammad Kachuee is approved.

Guy Van den Broeck

Yizhou Sun

Adnan Youssef Darwiche

Majid Sarrafzadeh, Committee Chair

University of California, Los Angeles

2020

To my parents . . .

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to thank Professor Majid Sarrafzadeh for his guidance in every part of this journey. His kind support and motivation not only made this work possible but also taught me what I believe will guide my life beyond PhD. Also, I would like to express my deep appreciation to the doctoral committee Professor Adnan Darwiche, Professor Yizhou Sun, and Professor Guy Van den Broeck for their constructive feedback and invaluable advice. Lastly, I want to thank my dear friends and members of the UCLA eHealth and Data Analytics Research Lab who helped me during this time. I greatly value these friendships and collaborations, and I believe that our connection will extend beyond this period.

VITA

2010–2014 B.Sc. of Electrical Engineering, Sharif University of Technology

2014–2016 M.Sc. of Electrical Engineering, Sharif University of Technology

PUBLICATIONS

M. Kachuee, K. Karkkainen, O. Goldstein, S. Darabi, M. Sarrafzadeh, Generative Imputation and Stochastic Prediction, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

S. Darabi, **M. Kachuee**, S. Fazeli, M. Sarrafzadeh, TAPER: Time-Aware Patient EHR Representation, *IEEE Journal of Biomedical and Health Informatics (JBHI)*, 2020.

O. Goldstein, **M. Kachuee**, K. Krkkinen,M. Sarrafzadeh, Target-Focused Feature Selection Using Uncertainty Measurements in Healthcare Data, *ACM Transactions on Computing for Healthcare*, 2020

M. Kachuee, O. Goldstein, K. Krkkinen, S. Darabi, M. Sarrafzadeh, Opportunistic Learning: Budgeted Cost-Sensitive Learning from Data Streams, *International Conference on Learning Representations (ICLR)*, 2019.

M. Kachuee, S. Darabi, S. Fazeli, M. Sarrafzadeh, Group-Connected Multilayer Perceptron Networks, *arXiv:1912.09600*, 2019.

M. Kachuee, K. Karkkainen, O. Goldstein, D. Zamanzadeh, M. Sarrafzadeh, Cost-Sensitive Diagnosis and Learning Leveraging Public Health Data, *arXiv:1902.07102*, 2019.

M. Kachuee, S. Darabi, B. Moatamed, M. Sarrafzadeh, Dynamic Feature Acquisition Using Denoising Autoencoders, *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2018.

M. Kachuee, S. Fazeli, M. Sarrafzadeh, ECG Heartbeat Classification: A Deep Transferable Representation, Accepted for publication in *IEEE International Conference on Healthcare Informatics (ICHI)*, 2018.

M. Kachuee, A. Hosseini, B. Moatamed, S. Darabi, M. Sarrafzadeh, Context-Aware Feature Query to Improve the Prediction Performance, *IEEE GlobalSIP Symposium on Big Data Analytics for IoT Healthcare (GlobalSIP),* 2017.

B. Moatamed, S. Darabi, M. Gwak, **M. Kachuee**, C. Metoyer, M. Linn, M. Sarrafzadeh Sport Analytics Platform for Athletic Readiness Assessment, *IEEE-NIH 2017 Special Topics Conference on Healthcare Innovations and Point-of-Care Technologies (HI-POCT),* 2017.

A. Esmaili, **M. Kachuee**, M. Shabany, Nonlinear Cuff-less Blood Pressure Estimation of Healthy Subjects Using Pulse Transit Time and Arrival Time, *IEEE Transactions on Instrumentation and Measurement (TIM)*, 2017.

M. Kachuee, L. Moore, T. Homsey, H. G. Damavandi, B. Moatamed, A. Hosseini, R. Huang, J. Leiter, D. Lu, M. Sarrafzadeh, An Active Learning Based Prediction of Epidural Stimulation Outcome in Spinal Cord Injury Patients Using Dynamic Sample weighting, *IEEE International Conference on Healthcare Informatics (ICHI)*, 2017.

CHAPTER 1

Introduction

In traditional machine learning settings, it is usually assumed that a training dataset is freely available and the objective is to train models that generalize well. In this paradigm, the feature set is fixed, and we are dealing with complete feature vectors accompanied by class labels that are provided for training. However, in many real-world scenarios, there are certain costs for acquiring features as well as budgets limiting the total expenditure. Here, the notation of cost is more general than financial cost and it also refers to other concepts such as computational cost [BET08, KKM17], privacy impacts [PKU15], energy consumption [KYR11], patient discomfort in medical tests [SS95, KMH17], user disruptions in computer and user interactions [EMF16], and so forth. Take the example of the disease diagnosis based on medical tests. Creating a complete feature vector from all the relevant information is synonymous with conducting many tests such as MRI scan, blood test, etc. which would not be practical. On the other hand, a physician approaches the problem by asking a set of basic easy-to-acquire features, and then incrementally prescribes other tests based on the current known information (i.e., context) until a reliable diagnosis can be made.

To overcome these issues, there are methods suggested in the literature trying to adapt feature selection algorithms to consider the cost of each feature [LXL17, GAS15, MHZ14, CZZ13]. However, another point of concern that requires attention is that selecting a fixed set of features to be used during the training phase and using them at test-time would not be an optimal solution; as it neglects the potential interdependence between features. In many scenarios, there are features that are either freely available or easy to acquire at test-time. An optimal decision about other features to include in the analysis can be highly dependent on them. For instance, a doctor decides whether to prescribe an MRI scan based on the

patient's current available information such as age, gender, symptoms and so on. In this example, having a fixed list of required tests and asking patients to provide the results of these tests for a clinical visit, would result in the high cost of MRI for all patients. In other words, the decision to include each feature should be based on the learned system dynamics as well as the available information at test-time.

In this manuscript, we suggest a novel approaches for cost-sensitive and context-aware feature acquisition and prediction at test-time. The proposed solutions are capable of incrementally asking for features to be included in the prediction based on the available context and user-defined feature costs. Also, as the cost-sensitive feature acquisition entails dealing with incomplete data, as a part of this work, we suggest a method for imputing missing values and measuring prediction uncertainties arising from missing values.

The rest of this manuscript is organized as follows. Chapter 2 reviews the current relevant literature. Chapter 3 introduces an approach for context-aware cost-sensitive learning based on sensitivity analysis in autoencoder architectures. Chapter 4 suggests an alternative solution based on deep reinforcement learning that is applicable to online stream processing settings. Chapter 5 presents an approach based on generative adversarial networks to impute missing values and estimate prediction uncertainties for incomplete data settings. Finally, Chapter 6 concludes the thesis.

CHAPTER 2

Related Work

2.1 Cost-Sensitive Learning

2.1.1 Static Feature Selection

One of the approaches to incorporate feature acquisition costs or feature costs in general is considering the feature costs during the training phase and trading off the prediction accuracy with the prediction cost. An example of these approaches is limiting the number of features that are actually used in the predictor model by using L_1 regularization [EHJ04]. In this method, the L_1 regularization enforces weights corresponding to certain features to be zero, and hence they can be omitted during the test phase. There are other methods in the literature that try to define and solve optimization problems over both the prediction performance and prediction costs [GGR02, GAS15, JC07]. Nevertheless, in all these methods, the final set of selected features is fixed and they fail to capture and take the advantage of the contextual information available at test-time.

2.1.2 Probabilistic Approaches

Alternatively, probabilistic methods were suggested that measure the value of each feature based on the current evidence. For instance, Choi *et al.* [CXD12] introduced same decision probability (SDP) as a criterion to measure the value of new evidence (i.e., feature) in terms of the probability of changing the current decision after observing that evidence. Chen *et al.* [CCD13, CCD14] continued that line of work suggested exact and approximate algorithms for computing SDP.

However, these methods are usually applicable to Bayesian networks or similar probabilistic models and make limiting assumptions such as having binary features and binary classes [CCD14]. Furthermore, these probabilistic methods are computationally expensive (usually of the complexity of PP^{PP}) and intractable for large scale problems [CCD15].

2.1.3 Tree-Based Approaches

An intuitive approach to incorporate feature costs during the training phase, while considering the available context during the test phase, is using the idea of decision trees. One of the most famous examples of this approach is the face detection cascade classifier by Viola and Jones [VJ04]. While their goal was to increase the prediction speed by rejecting negative samples as soon as possible within a cascade of classifiers, many papers followed their architecture and incorporated feature cost in creating cascade predictors [CXW12, XWC12]. One main drawback of cascade approaches is that cascades are only applicable to problems with a considerable class imbalance such as face detection or spam email detection. In these cases, the number of negative samples is significantly higher than the number of positive samples. However, there are many real-world applications in which the classes are relatively balanced such as document classification or image classification. To overcome this issue, in [XKW14, KBF12] authors suggested the idea of classifier trees instead of classifier cascades to handle the problems where cascades are not applicable.

While cascade and tree based test-time feature acquisition methods are shown to perform reasonably well in many scenarios; there are many problems and applications such as large-scale image classification, voice recognition, natural language processing, etc. where tree and cascade classifiers are not intrinsically strong enough to make accurate predictions. Another important limitation of cascade and tree based approaches is, while they include the context information to some extent, their feature query decisions are not truly instance specific. Specifically, they are limited by the fixed predetermined structure of the tree that enforces the features to be acquired at each tree node. A recent work by Nan *et al.* [NS17] suggested a gating method that employs adaptive linear or tree-based classifiers, alternating between low-

cost models for easy-to-handle instances and higher-cost models to handle more complicated cases. While this method outperforms many of the previous work on the tree-based and cascade cost-sensitive classifiers, the low-cost model being used is limited to simple linear classifiers or pruned random forests.

2.1.4 Sensitivity-Based Approaches

An alternative idea for measuring the informativeness of features given the context is using sensitivity analysis at test-time to measure the influence of each feature on the predictions. Early *et al.* [EFM16] introduced a method based on sensitivity analysis that exhaustively measures the impact of acquiring each feature on the prediction outcome. Their solution does not require training any other model, and it works in conjunction with almost any supervised learning algorithm. However, exhaustive sensitivity measurement is computationally expensive. It is impractical in problems with a large number of features to exhaustively examine the sensitivity with respect to each unknown feature.

In Chapter 3 we suggest a novel approach that is based on the idea of sensitivity analysis. The proposed approach incrementally asks for features based on the feature acquisition costs and the expected effect each feature can induce on the prediction. Furthermore, the devised method uses back-propagation of gradients and binary representation layers in neural networks to address the computational load as well as scalability concerns [KHM17, KDM18].

2.1.5 Imitation and Reinforcement Learning Approaches

Recently, there has been great attention toward using learning methods to solve the generic problem of cost-sensitive and context-aware feature acquisition. He *et al.* [HDE12] suggested a method based on imitation learning that trains a model that is able to predict an optimal feature query decision to be made given the available features. Contardo *et al.* [CDA16a, CDA16b] introduced the idea of defining the problem as a reinforcement learning problem and solving it as a separate problem. While these methods are successful in terms of truly incorporating the test-time context information to the decisions, they require extra effort of

training a feature query model in addition to the target predictor.

Imitation and reinforcement learning approaches are promising in terms of performance and scalability. However, the value functions used in these methods are usually not intuitive and require tuning hyper-parameters to balance the cost vs. accuracy trade-off. More specifically, they often rely on one or more hyper-parameters to adjust the average cost at which these models operate. On the other hand, in many real-world scenarios it is desirable to adjust the trade-off at the prediction-time rather than the training-time. For instance, it might be desirable to spend more resources for a certain instance or continue the feature acquisition until a desired level of prediction confidence is achieved [KGK19].

In Chapter 4, we propose a reinforcement learning based solution which is using novel reward function that measures the expected variations of prediction uncertainty after asking for each feature. Furthermore, the proposed method is applicable to stream learning settings where feature acquisition, learning, and prediction take place in an online manner [KGK19, KKG19].

2.2 Learning from Incomplete Data

2.2.1 Traditional Approaches

One of the simplest traditional methods for handling missing values includes imputing the occurrences of missing values with constant values such as zeros or using mean values. To enhance the accuracy of such imputations, alternatives such as k-nearest neighbors (KNN) [HTS99] and maximum likelihood estimation (MLE) [And57] have been suggested to estimate values to be used given an observed context. While these methods are easy to implement and analyze, they often fail to capture the complex feature dependencies as well as structures present in many problems.

2.2.2 Multiple Imputation

Rubin *et al.* [Rub76] suggested a categorization for missingness mechanisms: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). In this classification, a variable is MCAR if the occurrences of missing values do not depend on any other variable. If the occurrence of missing values is only dependent on observed variables it is classified as MAR, and if it is dependent on both observed and missing variables it is classified as MNAR. Under the assumption of MAR, the authors suggested multiple imputation (MI) as a stochastic imputation method. Here, instead of imputing missing values using a single value, several values are sampled to represent the distribution over the missing value. MI generates a few imputed complete datasets that are then used independently in statistical modeling [SG02, LR19]. Recent work by Aleryani *et al.* [AWD20] trains an ensemble of classifiers using bagging and stacking techniques based on multiple imputation of dataset samples, and studies the variance of the predictions made by each classifier.

Usually, the final goal of MI is to measure the robustness of the final statistical analysis amongst the imputed datasets. In other words, it measures the quality of imputations and the statistical significance of analysis on the imputed data. It should be noted that the number of imputations used in MI is usually very limited. Also, often strong simplifying assumptions are made in modeling the data distribution (e.g., multi-variate Gaussian or Student's t distribution) which limit the applicability of this method [SG02, Mur18].

2.2.3 Solutions based on Autoencoders

More recently, autoencoder architectures have been suggested as powerful density estimators capable of capturing complex distributions. Perhaps, denoising autoencoders (DAE) [VLB08, RKK20] are one of the most intuitive approaches in which a neural network is trained to reconstruct and denoise its input.

Following a more probabilistic perspective, variational autoencoders (VAE) [KW13] try to learn the data generating distribution via a latent representation. Specifically, conditional variational autoencoders (CVAE) [SLY15] can be used to sample missing values conditioned

on observed values. For instance, Mattei *et al.* [MF18] suggested a method based on deep latent variable models and importance sampling that offers a tighter likelihood bound compared to the standard VAE bound. While these methods are powerful generative models applicable to missing data imputation, often samples generated using autoencoders are biased toward the mode of the distribution (e.g., resulting in blurry images, for vision tasks) [GPM14, DBP16, SKS20].

2.2.4 Solutions based on Generative Adversarial Networks

Recently, due to the success of generative adversarial networks (GAN), there has been great attention toward applying them to impute missing values. For instance, Yoon *et al.* [YJV18] suggested an imputation method based on adversarial and reconstruction loss terms. Li *et al.* [LJM19] introduced the idea of using separate generator and discriminator networks to learn the missing data structure and data distribution. These methods have been quite successful and are able to present the state-of-the-art results. Though it should be noted that often the presence of additional loss terms may bias the generated samples toward the mode of the distribution being modeled. Also, these methods are often complicated to be applied in practical setups by practitioners. For instance, Yoon *et al.* [YJV18] requires setting hyperparameters to adjust the influence of an MSE loss term as well as the rate of discriminator hint vectors. Also as another example, Li *et al.* [LJM19] uses three generators and three discriminators for the final imputer architecture.

2.2.5 Impact of Missing Values on the Prediction Certainty

From the perspective of supervised analysis, imputation and handling missing values are usually considered as a preprocessing step. A few exceptions exist such as Bayesian models and decision trees that permit direct analysis on incomplete data [NJ09, ZQL05]. Note that while certain Bayesian methods such as probabilistic Bayesian networks allow handling of missing values as unobserved variables. However, given an incomplete training dataset and without any known causal structure as a priori, learning such models is a very challenging

problem with the complexity of at least NP-complete to learn the network architecture in addition to an iterative EM optimization to learn model parameters [Dar09, Nea04].

Tran *et al.* [TZA17] suggested a genetic programming method using multiple imputation to train a set of classifiers covering different combinations of observed features. While this method does not require any imputation at the prediction phase, it has significant limitations in the scale of the problems (i.e. the number of features/samples) that can be addressed due to the often combinatorial number of classifiers required. The recent work by Khosravi *et al.* [KCL19] suggests carefully designed probabilistic circuits to enable tractable inference given samples from a generative probabilistic model. Note that their method imposes certain structural constraints on the generative and discriminative networks. In [KVC20] authors suggest a method for probabilistic inference in decision tree models. They estimate the certainty of class assignments (expected prediction) based on partial assignment probabilities computed for the tree leaves. While it is a very promising line of research, using decision trees and requiring polynomial compute time is quite restrictive.

We argue that the simplistic approach of imputing missing values as a preprocessing step discards uncertainties that exist in original incomplete data samples. Instead, there is a need for methods that reflect these uncertainties on the final predicted target distribution. Chapter 5 suggests the idea of training a predictor on different imputed samples to capture the uncertainties over class assignments. Compared to MI, the suggested method interleaves imputation and training a downstream prediction model, enabling to estimate classification uncertainties for each instance.

CHAPTER 3

Cost-Sensitive Feature Acquisition Based on Sensitivity Analysis[1]

3.1 Problem Definition

We consider the problem of predicting target classes ($\boldsymbol{y} \in R^r$) corresponding to a given feature vector ($\boldsymbol{x} \in R^d$). Each feature vector consists of known features as well as unknown features (i.e., missing values) that are set to zero. The complete feature vector without any missing values is denoted by $\tilde{\boldsymbol{x}}$. To indicate unknown features, a vector $\boldsymbol{k} \in \{0,1\}^d$ is defined that acts as a mask and indicates known and unknown features with one and zero values, respectively. In addition, we define a feature acquisition cost vector ($\boldsymbol{c} \in R^d$) that defines the cost of acquiring each feature.

For simplicity of analysis, we consider the incremental problem of having a feature vector ($\boldsymbol{x}^t$) and the corresponding mask vector ($\boldsymbol{k}^t$) at time step t. Additionally, we consider the cost values to be time dependent and defined for each time step ($\boldsymbol{c}^t$). Using this notation, at each time step t, the current feature vector can be represented as

$$x_j^{(t)} = \begin{cases} 0 & k_j^t = 0 \\ \tilde{x_j} & k_j^t = 1 \end{cases}, \tag{3.1}$$

which is acquired at the total cost of

$$C_{total}^t = (\boldsymbol{k}^t - \boldsymbol{k}^0)^T \boldsymbol{c}^t. \tag{3.2}$$

[1]**This chapter is based on "M. Kachuee, S. Darabi, B. Moatamed, M. Sarrafzadeh, Dynamic Feature Acquisition Using Denoising Autoencoders, *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2018." ©2018 IEEE**

Apart from this, at each time step, we have an expected prediction value ($\boldsymbol{y}^t$) using a predictor function (h) that takes $\boldsymbol{x}^t$ as input:

$$\boldsymbol{y}^t = h(\boldsymbol{x}^t) = h(x_1^t, x_2^t, \ldots, x_d^t) \ . \tag{3.3}$$

In this setup, we define the feature query operator (q) as a function that acquires the value of feature j in the incomplete feature vector $\boldsymbol{x}^t$ and outputs the feature vector of the next time step, $\boldsymbol{x}^{t+1}$:

$$\begin{gathered} \boldsymbol{x}^{t+1} = q(\boldsymbol{x}^t, j), \text{where} \\ k_j^{t+1} - k_j^t = 1 \text{ and } k_i^{t+1} - k_i^t = 0 \ (i \neq j) \ . \end{gathered} \tag{3.4}$$

Furthermore, we define the desired feature to be queried at time t as the feature that decreases the prediction error significantly, while at the same time, incurs a low acquisition cost. Mathematically speaking, we can use prediction accuracy improvement per acquisition cost as a measure of efficiency for the feature query. Accordingly, the desired feature to be queried at time step t can be found by

$$j_{sel}^t = \underset{j \in \{1 \ldots d\} | k_j^t = 0}{argmin} \ |\tilde{\boldsymbol{y}} - h(q(\boldsymbol{x}^t, j))| \ . \ c_j^t \ , \tag{3.5}$$

where $\tilde{\boldsymbol{y}}$ is the ground-truth target value. It is worth mentioning that the solution introduced here is basically an incremental solution that greedily selects features to be acquired at each step. Table 3.1 presents a summary of the notations used throughout this chapter.

3.2 Sensitivity-based Feature Acquisition

While (3.5) suggests what features to be acquired at each step, directly using this equation is not practical. The reason behind this is the first term in this equation is usually not known and is difficult to estimate. To resolve this issue, it is possible to use the sensitivity of model predictions with respect to each missing feature as a measure for the potential impact of that feature on the final predictions. As a result, (3.5) can be rewritten using the suggested

Table 3.1: Summary of the notations used throughout this chapter.

Notation	**Description**
$\tilde{\boldsymbol{y}} \in R^r$	Ground-truth target values
$\boldsymbol{y} \in R^r$	Predicted target values
$\boldsymbol{x} \in R^d$	Incomplete feature vector at test-time
$\tilde{\boldsymbol{x}} \in R^d$	Complete feature vector without any missing values
$\boldsymbol{x_{bin}} \in R^{d \times l}$	Binary representation of the feature vector
$\boldsymbol{x'} \in R^d$	Reconstructed feature vector
$\boldsymbol{x'_{bin}} \in R^{d \times l}$	Binary representation of the reconstructed feature vector
$\boldsymbol{k} \in \{0,1\}^d$	Mask vector indicating known and unknown features
$\boldsymbol{c} \in R^d$	Feature acquisition cost vector
$\boldsymbol{z} \in R^{d'}$	Encoded feature vector

sensitivity measure as:

$$j^t_{sel} = \underset{j\in\{1...d\}|k^t_j=0}{argmax} \frac{Sensitivity(h(\boldsymbol{x}^t), j)}{c^t_j} \quad . \tag{3.6}$$

Note that because a higher sensitivity is synonymous with a more informative feature to select, the argmin function in (3.5) is replaced by argmax. Furthermore, the prediction sensitivity with respect to input j can be defined as

$$\begin{aligned} Sensitivity(h(\boldsymbol{x}^t), j) &= \mathbb{E}_{x_j}(\frac{\partial h^t(\boldsymbol{x}^t)}{\partial x_j}) \\ &= \int |\frac{\partial h(\boldsymbol{x}^t)}{\partial x_j}|\, p(x_j|\boldsymbol{x}^t; h^t)\, dx_j \quad . \end{aligned} \tag{3.7}$$

In this equation, the first term corresponds to the derivative of the predictor function with respect to each missing feature. The second term is the probability of the j'th feature given the available context and model parameters. By substituting this into (3.6), the feature query criterion can be written as

$$j^t_{sel} = \underset{j\in\{1...d\}|k^t_j=0}{argmax} \frac{\int |\frac{\partial h(\boldsymbol{x}^t)}{\partial x_j}|\, p(x_j|\boldsymbol{x}^t; h^t)\, dx_j}{c^t_j} \quad . \tag{3.8}$$

Furthermore, the continuous integral in (3.8) can be approximated by a discrete summation:

$$j^t_{sel} = \underset{j\in\{1...d\}|k^t_j=0}{argmax} \frac{\sum_{x_j\in RS} |\frac{\partial h(\boldsymbol{x}^t)}{\partial x_j}|\, p(x_j|\boldsymbol{x}^t; h^t)}{c^t_j} \quad , \tag{3.9}$$

where RS is a set of samples from the range of possible values that can be taken by each feature. By adjusting the granularity of the values in the RS set, one can trade-off between the approximation accuracy and the computational load of the expected value approximation.

3.3 Proposed Solution

The required terms in (3.9) for finding the feature to query includes: the cost of acquiring each feature, the derivative of the prediction function with respect to each input at different input values, and probability of having each value for each feature given the available context. Feature query costs are assumed to be given by the user and known for each time step. For

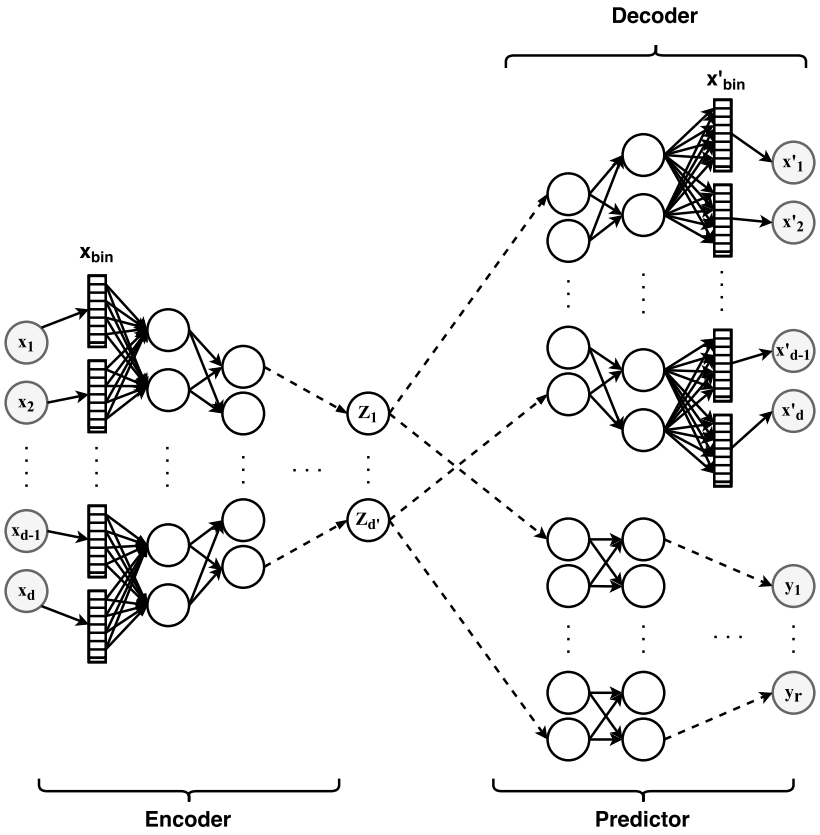


Figure 3.1: Network architecture of the proposed method including: encoder, decoder, and predictor parts. The encoder part is responsible for handling missing features. The decoder part is used for feature density estimation. The predictor is responsible for making predictions; additionally, its derivatives with respect to inputs are used for measuring sensitivities.

the latter two, while it is possible to model and estimate each term using conventional modeling methods, the solution to evaluate the summation exhaustively may be computationally expensive and impractical in many applications. Here, we introduce a novel method based on autoencoders with binary representation layers that can estimate the whole summation with a single forward and backward propagation in neural networks.

The left and the upper right part of the Figure 3.1 show the architecture of the proposed network for the context-aware and one-shot estimation of the distribution of each feature. As depicted, an autoencoder architecture designed to convert each feature in the feature vector ($\boldsymbol{x}$) to a binary representation ($\boldsymbol{x_{bin}}$). Then, it encodes the features to a more compact representation ($\boldsymbol{z}$), and finally reconstructs the original feature vectors ($\boldsymbol{x'}$) by creating a binary decoded vector ($\boldsymbol{x'_{bin}}$). Here, in order to have an estimate for the probability of each bit being set, sigmoid non-linearity activation function is used for the binary reconstruction layer ($\boldsymbol{x'_{bin}}$). For other activation functions; however, we used the rectified linear unit (ReLU) [NH10] non-linearity. Additionally, the network optimization cost function is defined as the weighted sum of cross-entropies for binary feature words. Here, the term word refers to the set of encoded bits that are representing a feature. The weights are adjusted to offset the importance of the reconstruction error caused by errors in different bits in the word with different significance. It is worth mentioning that the trained autoencoder as explained here, takes an input feature vector where missing features are set to zero, and it is capable of estimating the probability of each bit being set in the binary decode layer ($\boldsymbol{x'_{bin}}$).

In addition to the autoencoder part, in the network of Figure 3.1, we create a predictor model by stacking a few layers on the top of the encoded representation ($\boldsymbol{z}$) and training the encoder as well as the predictor parts of the network in a supervised fashion. Here, in order to measure the sensitivity of the output predictions with respect to different changes in each feature, we suggest using the summation of absolute derivatives of the output layer neurons with respect to each bit of the missing features. The final estimation of the summation in (3.9) is achieved by the inner product of the bit probabilities estimated from the autoencoder's binary reconstruction layer and the sensitivities calculated from the derivative of output layer with respect to each input feature bit. Specifically, we suggest defining the

RS as

$$RS = \{2^{-l}, \ldots, 2^{-2}, 2^{-1}, 1\} \ , \tag{3.10}$$

where l is the total number of bits used in the binary representation of each feature. Using (3.9) and (3.10), the feature to be acquired is given by

$$j^t_{sel} = \underset{j \in \{1 \ldots d\} | k^t_j = 0}{argmax} \frac{\sum_{b=1}^{b=l} |\frac{\partial h(\boldsymbol{x}^t)}{\partial x_{bin_{j,b}}}| \, x'_{bin_{j,b}}}{c^t_j} \ , \tag{3.11}$$

where the sensitivity term is defined as

$$|\frac{\partial h(\boldsymbol{x}^t)}{\partial x_{bin_{j,b}}}| = \sum_{i=1}^{i=r} |\frac{\partial y_i}{\partial x_{bin_{j,b}}}| \ . \tag{3.12}$$

It is worth mentioning that, in addition to the common neural network hyper-parameters, the only hyper-parameter that is added by the suggested method is the parameter l which is used for controlling the accuracy of the binary representation. Additionally, as we do not make any assumptions on the values used as feature costs and the proposed method is incremental, it can be applied to the scenarios where feature costs are subject to change during the course of operation at test-time. However, in our experiments, in order to make the comparison of results easier, we evaluate the proposed method on scenarios where feature acquisition costs are constant in time.

3.4 Implementation Details

Prior to the analysis we have normalized all feature values in the dataset to the range of zero and one. Also, throughout the experiments we used Tensorflow numerical computation library [AAB16] and explored feed-forward neural network architectures. Also, ReLU non-linearity [NH10] is used for all hidden layers except the binary representation layers. For converting feature values to the suggested binary representation, we implemented the bit by bit recursive conversion in an efficient and parallel manner. Also, for converting back from the binary representation, we implemented the weighted summation of bit values utilizing a fully parallel matrix multiplication, reshape, and addition. In this work, the Adaptive Moment (Adam) optimization algorithm [KB14] is used to train each network. The Adam

hyper-parameters: learning rate (α), decay rate for the first moment (β_1), and decay rate for the second raw moment (β_2) are set to 0.001, 0.9, and 0.999, respectively.

The process of training the network starts with training the autoencoder part using a weighted cross-entropy loss between the binary representation of the complete feature vectors and the estimated probabilities from the binary reconstruction layer:

$$f(x,\theta) = \sum_{j=1}^{d} \sum_{b=0}^{l} 2^{-b} \left(\tilde{x}'_{bin_{j,b}} log(x'_{bin_{j,b}}) + (1 - \tilde{x}'_{bin_{j,b}}) log(1 - x'_{bin_{j,b}}) \right) . \tag{3.13}$$

In order to train the denoising autoencoder, for each training instance, we sample random values from a latent Beta distribution and use the sampled values as the probability of missing each feature in the training data. After training the autoencoder part, the trained autoencoder network weights are stored, and a few prediction layers are added on top of the encoder part. The reason we store autoencoder weights is that fine-tuning the weights for the prediction task would affect the distribution estimation functionality of the originally trained autoencoder. In other words, we do fine-tuning for the supervised prediction task, while a copy of the original not fine-tuned autoencoder is used for probabilistic modeling. Here, to train the predictor network, we use a smaller learning rate ($\alpha = 0.0001$) for the pre-trained encoder and a larger learning rate ($\alpha = 0.001$) for the new predictor weights.

For the efficient calculation of derivatives we use back-propagation from the values in the output prediction layer to each binary input bit. In this section, we only described the general architecture and training procedures, as we have conducted various experiments on different datasets, the exact network architecture of each case is explained in Chapter 3.5.

3.5 Experimental Results

3.5.1 Datasets and Experiments

The proposed method is evaluated on seven different real-world datasets including human activity recognition (HAPT) [ROS16], hand-written character recognition (MNIST) [LCB98],

Table 3.2: The summary of datasets and experimental settings.

Dataset	Instances	Features	Classes	Network Architecture	Latent Missing Distribution
MNIST	70000	784 [a]	10	Encoder: [697×8, 64, 32]	Beta Distribution
[LCB98]				Predictor: [16,10]	$\alpha = 3.5, \beta = 1.5$
Yahoo LTRC	34815	519	5	Encoder: [519×8, 128, 32]	Beta Distribution
[CC11]				Predictor: [16, 8]	$\alpha = 1.5, \beta = 1.5$
HAPT	10929	561	12	Encoder: [561×8, 64, 32]	Beta Distribution
[ROS16]				Predictor: [16]	$\alpha = 1.5, \beta = 1.5$
Reuters R8	7674	1000	8	Encoder: [1000×8, 64, 32]	Beta Distribution
[Lew97]				Predictor: [16, 8]	$\alpha = 3.5, \beta = 1.5$
UCI Mushroom	8124	22 [b]	2	Encoder: [116×8, 16]	Beta Distribution
[Sch81]				Predictor: [4]	$\alpha = 5.5, \beta = 1.5$
UCI Landsat	6435	36	7	Encoder: [36×8, 16, 8]	Beta Distribution
[BM98]				Predictor: [4]	$\alpha = 1.5, \beta = 1.5$
UCI CTG	2126	23	3	Encoder: [23×8, 8]	Beta Distribution
[CBG00]				Predictor: [4]	$\alpha = 1.5, \beta = 1.5$
Synthesized	16000	64	2	Encoder: [64×8, 16, 10]	Beta Distribution
(Section 3.5.4)				Predictor: [8, 4]	$\alpha = 1.5, \beta = 1.5$
Thyroid	279	16	3	Encoder: [16×8, 8]	Beta Distribution
(Section 3.5.5)				Predictor: [4]	$\alpha = 1.5, \beta = 1.5$

[a] 697 features after omitting features with STD of less than 0.0001 corresponding to margin pixels.

[b] 116 features after one-hot encoding of categorical features.

document classification (Reuters R8) [Lew97], and web ranking (Yahoo LTRC) [CC11] as well as three other classification datasets. Apart from these, we have evaluated the method on a synthesized dataset which is explained in Section 3.5.4 and a dataset in health domain explained in Section 3.5.5. Table 3.2 presents a summery of the conducted experiments. The table also includes the network architecture and the missing value distribution used during the training phase. In each case, the architecture column contains encoder layer sizes for the binary layers, encoder layers, and predictor layers. In this table, the decoder layers are not shown and are equal to the encoder layer sizes in reverse order. We used an 8-bit binary representation throughout the experiments. Regarding the feature size of each dataset, we report both the nominal feature count and the number of features we have used in our experiments. Specifically, for the MNIST dataset, each pixel is considered as a feature and we removed features corresponding to pixels near the margin that are almost always zero with the standard deviation of less than 0.0001 across all samples. Also, regarding the Mushroom dataset, one-hot encoding of 22 categorical features resulted in 116 features to be used as input. An important point to consider for these features is that, during sensitivity measurement and acquisition, we should consider acquisition of all one-hot features corresponding to a categorical feature as a single feature to acquire.

Regarding the feature acquisition costs, for the LTRC dataset, as suggested by [XKW14], we used real cost values in the range of 1 to 150 based on the time required to extract each feature. In order to introduce feature costs to the MNIST dataset, we have followed the method suggested by [TS13]. Using this method, we create feature vectors by concatenating MNIST images at different resolutions including 28×28, 14×14, 7×7, and 4×4 with feature acquisition costs of 4, 3, 2, and 1 for acquiring feature at each resolution, respectively. For the Landsat dataset, each sample consists of features from four different frequency bands. Here, we considered features of the same frequency band to have the same acquisition cost from 1 to 4, equal to the frequency band number. At last, for the CTG dataset, we assumed that features that are measuring event counts to have cost value of 1, features measuring statistical information to have the cost value of 2, and histogram features to have the cost of 3. For the synthesized and diabetes datasets, a complete explanation experiments is presented in

Table 3.3: Results of evaluating the proposed method on different datasets.

Dataset	FACT Accuracy (%)					RFC Accuracy	Denoising	Rand	DPFQ[3]	FACT
	% of total cost used[1]					(%)	(%)	(AUACC[2])	(AUACC)	(AUACC)
	0%	25%	50%	75%	100%	100% total cost[1]				
MNIST	10.63	94.95	96.19	96.17	96.17	97.07	60.08	0.63	0.71	0.82
Yahoo LTRC	20.70	47.16	49.41	50.48	50.39	50.51	94.03	0.47	0.47	0.48
HAPT	16.60	87.57	90.22	90.71	90.77	91.75	90.09	0.65	0.69	0.76
Reuters R8	50.74	94.78	95.56	94.78	94.70	94.61	14.92	0.87	0.89	0.93
UCI Mushroom	50.41	99.26	99.83	99.91	99.93	99.99	57.69	0.61	0.85	0.85
UCI Landsat	27.66	80.10	83.73	87.35	88.49	91.08	93.88	0.65	0.68	0.72
UCI CTG	74.52	84.90	88.68	88.99	90.88	89.62	77.48	0.83	0.84	0.86
Synthesized	50.43	96.72	98.19	98.25	98.33	99.41	88.78	0.64	0.73	0.78
Thyroid	26.83	65.85	70.73	75.61	78.05	80.85	44.07	0.59	0.59	0.71

[1] Total cost defined as the total cost of acquiring all features.

[2] AUACC is defined as the area under the accuracy and the feature acquisition cost curve.

[3] Cost-aware version of the method suggested in [KHM17].

Section 3.5.4 and Section 3.5.5, respectively. Lastly, for the other three datasets, we assumed feature costs to be equal for all features.

Based on the aforementioned experimental setup, in the following parts of this section, we evaluate the proposed method for feature acquisition considering costs at test-time (FACT) on each dataset.

3.5.2 Performance Evaluation

We split each dataset into three parts: 15% for test, 15% for validation, and the rest for training. During the training phase, we use the validation and train sets to train the networks as explained in Section 3.4 and following the setups introduced in Table 3.2. It is worth noting that the proposed method does not necessary require all training features to be known. In fact, as explained in Section 3.4, we use a latent Beta distribution to simulate the existence

of unknown features. After the training phase, we use the test set to simulate the case of feature acquisition using the proposed method by assuming all the features to be initially unknown and using the feature query criterion of (3.11) to ask for features incrementally. In our experiments (except experiments in Section 3.5.4), we continue the feature query until querying all the features and report the accuracy as well as the cost at each point during feature acquisition. In real applications; however, the incremental acquisition should be stopped after reaching a certain criterion such as a minimum confidence of predictions.

Table 3.3 presents the results of the proposed method on each dataset. The table contains the test accuracy of each dataset while asking for different percentage of the total cost from the original feature set. Here, total cost is defined as the cost of acquiring all features. As a baseline, we have reported the test performance of using a random forest classifier (RFC) on the complete feature set (i.e., acquiring all features and spending the maximum cost). The table also reports the denoising percentage of the trained denoising autoencoder calculated as

$$100 \times \frac{||\boldsymbol{x} - \tilde{\boldsymbol{x}}|| - ||\boldsymbol{x'} - \tilde{\boldsymbol{x}}||}{||\boldsymbol{x} - \tilde{\boldsymbol{x}}||} \quad . \tag{3.14}$$

Additionally, in this table, the area under the accuracy-cost curves (AUACC) of the proposed feature query method as well as the AUACC of randomly asking for the unknown features are presented. We have also included AUACC results of a cost-aware version of the method suggested in [KHM17] in which sensitivities are normalized by feature costs (see the DPFQ column). It is worth mentioning that AUACC values are calculated as the normalized area under the accuracy versus the acquisition cost curve from a cost of zero to the cost at which the accuracy converges to the maximum accuracy.

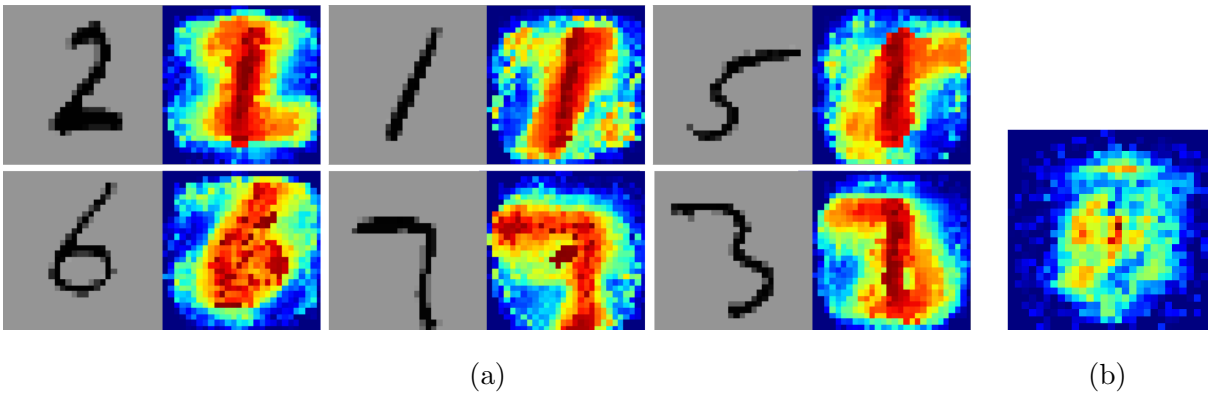
According to the results presented in Table 3.3, the proposed method can be used to effectively reduce the cost of features to be acquired at test-time for accurate predictions. Also, comparing the baseline accuracy of using the complete feature set, the results show that our method trains predictor models that can make viable predictions using only a subset of the features without sacrificing prediction performance. Regarding the denoising percentage, in most of the cases, the achieved denoising percentage is significant and confirming that a denoising autoencoder is capable of encoding features to reduce the feature representation

length which results in a new representation that is more robust to the presence of missing values. Regarding the area under the curve values, in all cases, the AUC values of using FACT is considerably higher than its random selection counterpart. It is also noteworthy to mention that for a few datasets (i.e., Reuters R8, UCI Landsat, HAPT, and UCI CTG) there is a considerable class imbalance that is affecting baseline accuracies.

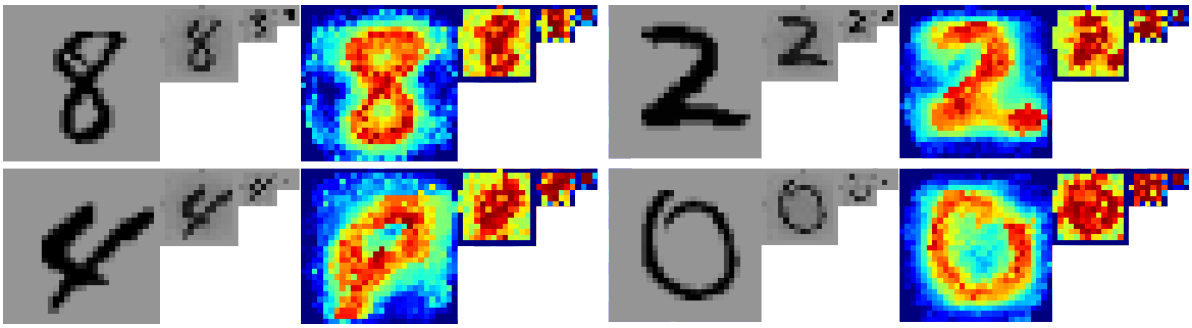
To further illustrate the performance of the proposed approach, we used MNIST dataset to visualize the effect of cost and context on the selection of features to be queried. Here, features are pixel values at different locations across each image, and the context is the available pixel values at each time step. Figure 3.2 shows the effect of context and cost on the order of features that are queried by the proposed algorithm as well as a static order which is measured based on mutual information between pixels and target classes. Here, we present results for the original MNIST dataset with single resolution images and equal feature costs (see Figure 3.2a) as well as the introduced multi-resolution setup with different feature costs for each resolution (see Figure 3.2c). In this figure, pixels with higher importance to be queried are indicated by warmer colors and less important pixels are indicated by colder colors. As it is evident from this figure, the proposed context-aware method, based on the available pixels, acquires features with different orders and is scanning for digit edges or discriminative areas. On the other hand, the static feature acquisition method, only asks for central pixels of each image in a fixed order (see Figure 3.2b). In addition, regarding the multi-resolution case, as it can be seen from the figure, the informative pixels from lower resolutions that incur lower costs are preferred to more costly higher resolution pixels. For instance, in the lower left corner image of Figure 3.2c, the parts that are first acquired from the high-resolution pixels are the parts that create difference between the digits of 4 and 9 which are not clear enough in lower resolutions.

3.5.3 Comparison with Other Work

Figure 3.3 presents comparison of the proposed feature acquisition method with a feature acquisition method based on recurrent neural networks (RADIN) [CDA16a] and a tree-



(a) (b)



(c)

Figure 3.2: Visualization of: (a) using the proposed approach on the MNIST dataset with equal feature costs, (b) using static feature acquisition using mutual information between pixels and targets on MNIST dataset, and (c) using the proposed approach on the multi-resolution MNIST dataset with different feature costs at each resolution. Pixels with more importance/priority to be queried are indicated by warmer colors.

based feature acquisition method (GreedyMiser) [XWC12]. In this comparison, we have used MNIST dataset to evaluate the performance of each method based on their accuracy using a different number of features. As it can be inferred from the figure, in the case of acquiring 10% of features, where the number of features to be queried is significantly less than the total number of features, the achieved accuracy using FACT is lower than other methods. Nevertheless, the rate of increase in the accuracy with respect to the number of queried features for the presented method is significantly higher than other papers which makes it superior in other cases. In this plot and similar cost versus accuracy plots in this section, we provide 95% confidence intervals presented as error-bars measured by running

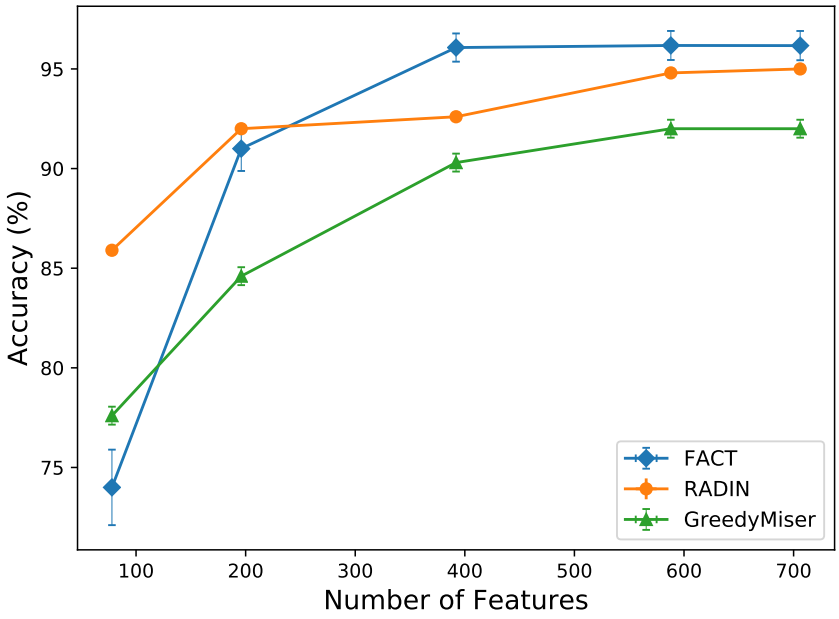


Figure 3.3: Comparison of the proposed method (FACT) with RADIN [CDA16a] and GreedyMiser [XWC12] methods on the MNIST dataset.

each experiment multiple times using different random initializations.

Figure 3.4 presents a comparison between the feature acquisition and cost curve of the proposed method with three other approaches that use classifier cascades or trees in the literature including CSTC [XKW14], Cronus [CXW12], and Early Exit [CZC10]. Here, we used the LTRC dataset with the feature costs as suggested by [XKW14] to plot the feature acquisition cost versus the corresponding normalized discounted cumulative gain (NDCG) [JK02] performance measure. NDCG is a well-known measure of ranking quality that is used to measure the effectiveness and the relevance of ranking results in search engines. As it can be seen from this figure, FACT is significantly more powerful and more efficient compared to others.

In order to evaluate the performance of the proposed method for the estimation of sensitivity values, we have implemented an exhaustive feature query method as suggested by [EFM16] using sensitivity as the utility function. To make a fair comparison, we have used the trained predictor network, and exhaustively measured the effect of changing each input

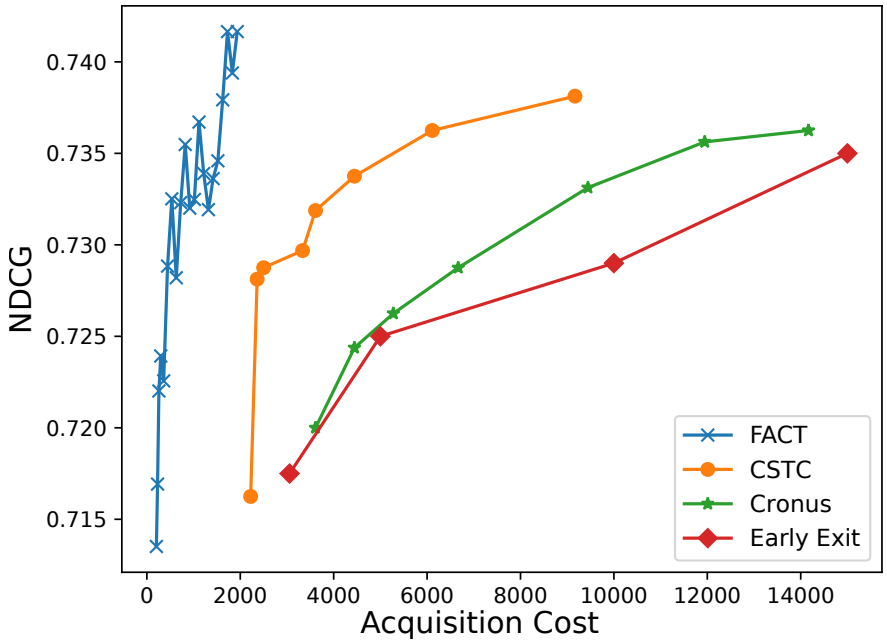


Figure 3.4: Comparison of the proposed method (FACT) with CSTC [XKW14], Cronus [CXW12], and Early Exit [CZC10] methods on the LTRC dataset.

on the prediction probabilities. Here, in order to estimate the probability of each change, we have used a 5-bin histogram for each feature. Figure 3.5 presents a comparison between the accuracy achieved using FACT and the exhaustive sensitivity-based method on the Landsat dataset. As a baseline, we have also included the curve corresponding to randomly selecting and acquiring features. As it is evident from the figure, the proposed method is almost equivalent to the exhaustive method in terms of the accuracy achieved at each total acquisition cost. This is promising considering the fact that the proposed approach tries to approximate the exhaustive sensitivity measurement in an efficient and scalable manner. In other words, compared to the proposed method, the exhaustive method is significantly slower and less efficient at test-time. Specifically, the average processing time of the exhaustive method was about 2400 ms for each sample, while the corresponding processing time for FACT was about 20 ms (i.e., about 120 times faster). In other test cases with more features, comparing with the exhaustive method was not possible due to the exponentially increasing computational load of evaluating the exhaustive method.

It worth mentioning that this computational advantage comes from the fact that, for

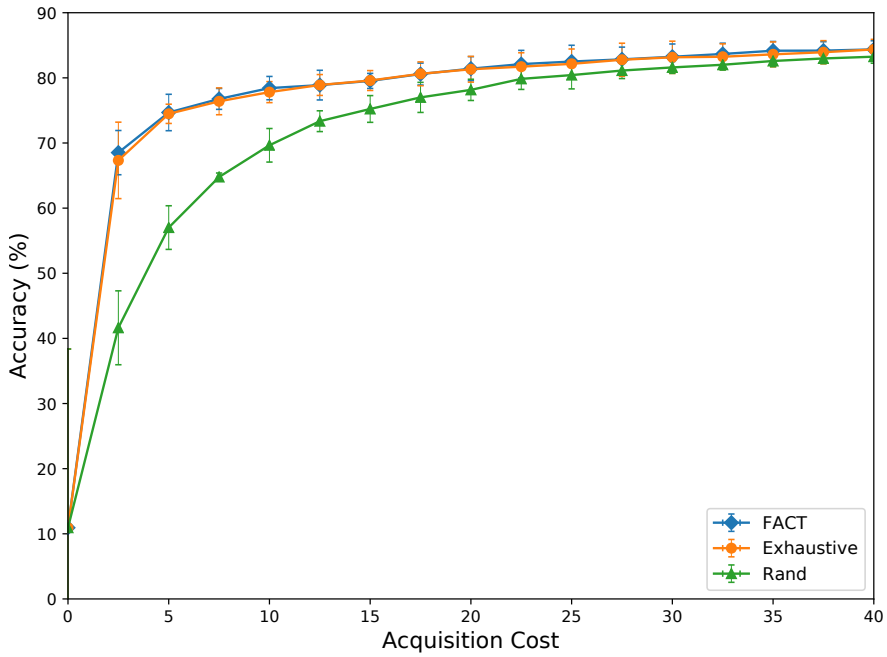
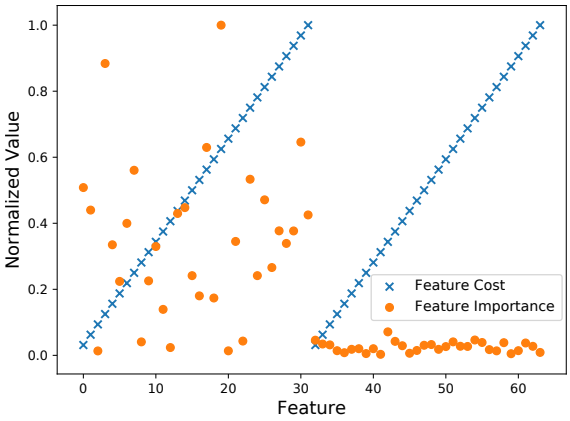


Figure 3.5: Comparison of the proposed method (FACT) with exhaustive sensitivity-based (Exhaustive) [EFM16] and random selection (Rand) methods on the Landsat dataset. This figure shows that the proposed method is able to approximate the ground-truth sensitivity values accurately and efficiently.

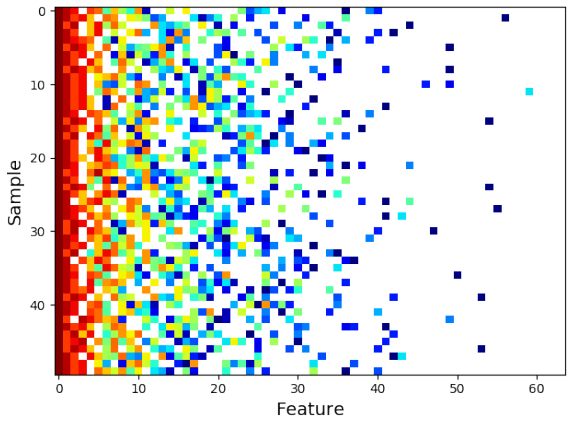
each incremental feature query, we approximate the summation of (3.11) for all unknown features using one forward and one backward network computation. However, the exhaustive sensitivity measurement computes the summation for each feature and over the range of all possible values, separately. In other words, the proposed method scales linearly with the growth in the number of unknown features, while the exhaustive method scales in polynomial time.

3.5.4 Evaluation using Synthesized Data

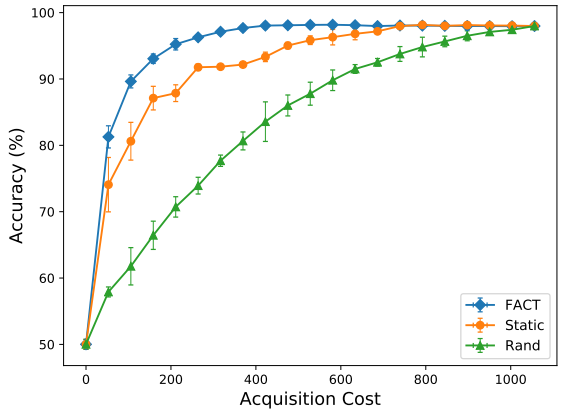
In order to get more insight about the performance of the proposed method, we have used a synthesized dataset to evaluate the suggested approach. The synthesized dataset is generated as follows: first, we have randomly sampled 16 cluster centers from a 32-dimensional space. Then, 1000 points sampled around each cluster center from a normal distribution



(a)



(b)



(c)

Figure 3.6: Evaluation of the proposed method on synthesized data. (a) Cost and static importance of each feature. (b) The feature acquisition order for 50 different test samples (warmer colors mean more priority). (c) Accuracy versus acquisition cost curves for the proposed method (FACT), acquisition using static order, and random selection.

with the mean of zero and variance of 0.25. Afterwards, we have randomly assigned each cluster to a class from a set of two different classes. To each feature vector created so far containing 32 features, we have appended another 32 features with random values from a normal distribution. These are features without any predictive value. Accordingly, the resulting feature vectors are of size 64. Finally, we made the dataset cost-sensitive by defining feature costs for the first and second 32 features to be a monotonically increasing function from 1 to 32. See Figure 3.6a for a visualization of feature costs and the static importance computed using the mutual information between features and labels.

Figure 3.6b demonstrates the order in which each feature is acquired using the proposed method. In this figure, each row corresponds to a test sample (here only 50 samples are visualized) and each column represents a feature. The features that are acquired earlier are indicated with warmer colors. Here, we have continued the feature acquisition until we reach 95% of the maximum achievable accuracy. As it can be seen, the second half of features which are not informative are mainly skipped by the proposed method. Specifically, only about 5.2% of features from the second half are selected by FACT, which means that most of the uninformative features are not acquired by the algorithm. On the other hand, based on the cost and value of each feature from the first half, the proposed method acquired features that are more informative and have lower cost values. Apart from this, Figure 3.6c presents the accuracy versus total feature acquisition cost on the test set. As it can be seen from the curve, FACT converges to the maximum accuracy much faster than the static and random acquisition methods. It is mainly due to the fact that the proposed method highly prefers informative features with low cost while other methods disregard this information.

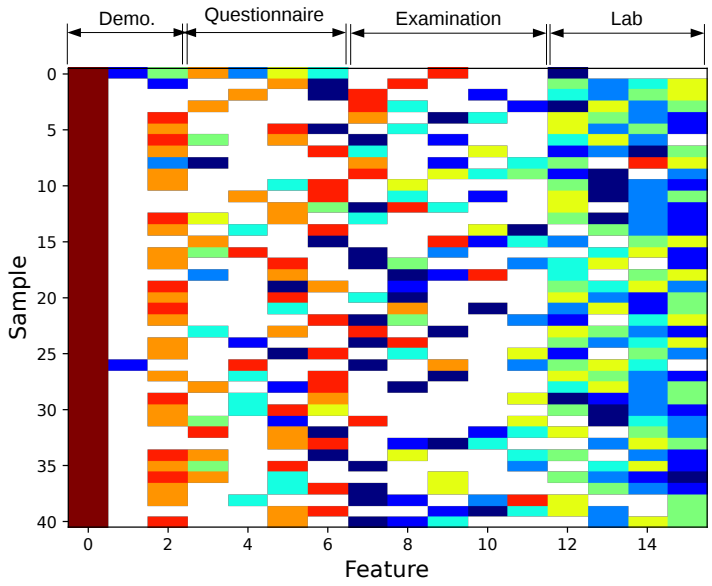
3.5.5 Evaluation using Real-World Health Data

In order to evaluate the performance of the proposed method on a dataset in health domain where feature acquisition costs are inherently important, we have used thyroid classification dataset [DK17][2]. Here, we have features from different categories including demographics,

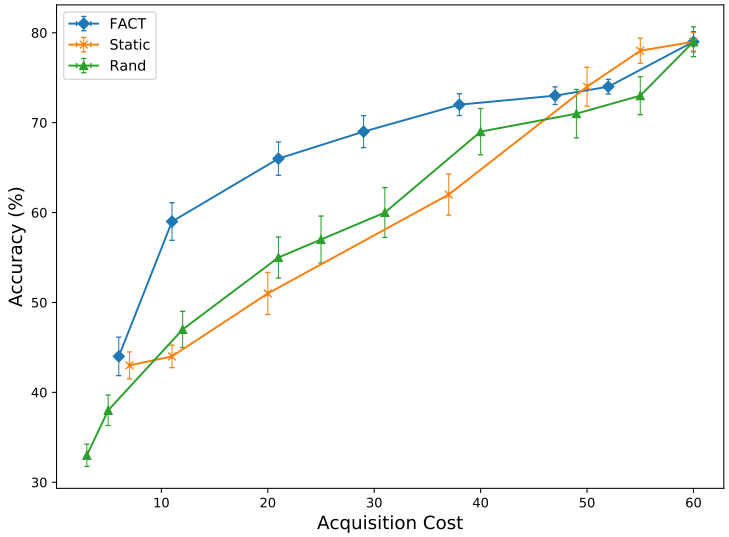
[2]Available at: http://archive.ics.uci.edu/ml/datasets/thyroid+disease

questionnaire, examination, and lab results. Furthermore, this dataset provides the acquisition costs corresponding to each feature which ranges from 1.0 for features such as age to 22.78 for certain blood tests.

Figure 3.7a presents a visualization of orders which each feature is acquired for 40 randomly selected test samples. As it can be observed from this visualization, FACT gives more priority to low cost features and costly but informative features are acquired a with lower priority. Apart from this, Figure 3.7b presents the accuracy versus acquisition cost curve for FACT, static, and random methods. As it can be seen from this figure, FACT outperforms other baseline approaches.



(a)



(b)

Figure 3.7: Evaluation of the proposed method on thyroid disease classification task. (a) The feature acquisition order for 50 different test samples (warmer colors mean more priority). (b) Accuracy versus acquisition cost curves for the proposed method (FACT), acquisition using static order, and random selection.

CHAPTER 4

Budgeted Cost-Sensitive Learning from Data Streams[1]

4.1 Preliminaries

4.1.1 Problem Settings

In this chapter, we consider the general scenario of having a stream of samples as input (S_i). Each sample S_i corresponds to a data point of a certain class in $\mathbb{R}^d$, where there is a cost for acquiring each feature ($\boldsymbol{c}_j; 1 \leq j \leq d$). For each sample, initially, we do not know the value of any feature. Subsequently, at each time step t, we only have access to a partial realization of the feature vector denoted by $\boldsymbol{x}_i^t$ that consists of features that are acquired so far. There is a maximum feature acquisition budget (B) that is available for each sample. Note that the acquisition may also be terminated before reaching the maximum budget, based on any other termination condition such as reaching a certain prediction confidence. Furthermore, for each S_i, there is a ground truth target label $\tilde{y}_i$. It is also worth noting that we consider the online stream processing task in which acquiring features is only possible for the current sample being processed. In other words, any decision should take place in an online fashion.

In this setting, the goal of an Opportunistic Learning (OL) solution is to make accurate predictions for each sample by acquiring as many features as necessary. At the same time, learning should take place by updating the model while maintaining the budgets. Please note that, in this setup, we are assuming that the feature acquisition algorithm is processing a stream of input samples and there are no distinct training or test samples. However, we

[1]**This chapter is based on "M. Kachuee, O. Goldstein, K. Krkkinen, S. Darabi, M. Sarrafzadeh, Opportunistic Learning: Budgeted Cost-Sensitive Learning from Data Streams, *International Conference on Learning Representations (ICLR)*, 2019."**

assume that ground truth labels are only available to us after the prediction and for a subset of samples.

More formally, we define a mask vector $\boldsymbol{k}_i^t \in \{0,1\}^d$ where each element of $\boldsymbol{k}$ indicates if the corresponding feature is available in $\boldsymbol{x}_i^t$. Using this notation, the total feature acquisition cost at each time step can be represented as

$$C_{total,i}^t = (\boldsymbol{k}_i^t - \boldsymbol{k}_i^0)^T \boldsymbol{c} \quad . \tag{4.1}$$

Furthermore, we define the feature query operator (q) as

$$\boldsymbol{x}_i^{t+1} = q(\boldsymbol{x}_i^t, j), \text{ where } \boldsymbol{k}_{i,j}^{t+1} - \boldsymbol{k}_{i,j}^t = 1 \quad . \tag{4.2}$$

In Section 4.2, we use these primitive operations and notations for presenting the suggested solution.

4.1.2 Prediction Certainty

As prediction certainty is used extensively throughout this chapter, we devote this section to certainty measurement. The softmax output layer in neural networks are traditionally used as a measure of prediction certainty. However, interpreting softmax values as probabilities is an ad hoc approach prone to errors and inaccurate certainty estimates [SZS13]. In order to mitigate this issue, we follow the idea of Bayesian neural networks and Monte Carlo dropout (MC dropout) [Wil97, GG16]. Here we consider the distribution of model parameters at each layer l in an L layer neural network as:

$$\hat{\omega}_l \sim p(\omega_l), \text{ where } 1 \leq l \leq L \quad , \tag{4.3}$$

where $\hat{\omega}_l$ is a realization of layer parameters from the probability distribution of $p(\omega_l)$. In this setting, a probability estimate conditioned on the input and stochastic model parameters is represented as:

$$p(y|\boldsymbol{x}, \hat{\omega}) = \text{softmax}(f_{\mathbb{D}}^{\hat{\omega}}(\boldsymbol{x})) \quad , \tag{4.4}$$

where $f_{\mathbb{D}}^{\hat{\omega}}$ is the output activation of a neural network with parameters $\hat{\omega}$ trained on dataset $\mathbb{D}$. In order to find the uncertainty of final predictions with respect to inputs, we integrate

(4.4) with respect to ω:

$$p(y|\boldsymbol{x}, \mathbb{D}) = \int p(y|\boldsymbol{x}, \omega) p(\omega|\mathbb{D}) d\omega \quad . \tag{4.5}$$

Finally, MC dropout suggests interpreting the dropout forward path evaluations as Monte Carlo samples ($\overline{\omega}_t$) from the ω distribution and approximating the prediction probability as:

$$p(y|\boldsymbol{x}, \mathbb{D}) = \frac{1}{T} \sum_{t=1}^{T} p(y|\boldsymbol{x}, \overline{\omega}_t) \quad . \tag{4.6}$$

With reasonable dropout probability and number of samples, the MC dropout estimate can be considered as an accurate estimate of the prediction uncertainty. Readers are referred to [GG16] for a more detailed discussion. In this chapter, we denote the certainty of prediction for a given sample ($Cert(\boldsymbol{x}_i^t)$) as a vector providing the probability of the sample belonging to each class in (4.6).

4.2 Proposed Solution

4.2.1 Cost-Sensitive Feature Acquisition

We formulate the problem at hand as a generic reinforcement learning problem. Each episode is basically consisting of a sequence of interactions between the suggested algorithm and a single data instance (i.e., sample). At each point, the current state is defined as the current realization of the feature vector (i.e., $\boldsymbol{x}_i^t$) for a given instance. At each state, the set of valid actions consists of acquiring any feature that is not acquired yet (i.e., $A_i^t = \{j = 1 \dots d | \boldsymbol{k}_{i,j}^t = 0\}$). In this setting, each action along with the state transition as well as a reward, defined in the following, is characterizing an experience.

We suggest incremental feature acquisition based on the value per unit cost of each feature. Here, the value of acquiring a feature is defined as the expected amount of change in the prediction uncertainty that acquiring the feature causes. Specifically, we define the value of each unknown feature as:

$$r_{i,j}^t = \frac{||Cert(\boldsymbol{x}_i^t) - Cert(q(\boldsymbol{x}_i^t, j))||}{\boldsymbol{c}_j} \quad , \tag{4.7}$$

where $r_{i,j}^t$ is the value of acquiring feature j for sample i at time step t. It can be interpreted as the expected change of the hypothesis due to acquiring each feature per unit of the cost. Other reinforcement learning based feature acquisition methods in the literature usually use the final prediction accuracy and feature acquisition costs as components of reward function [HMK16, SHY17, JPL17]. However, the reward function of (4.7) is modeling the weighted changes of hypothesis after acquiring each feature. Consequently, it results in an incremental solution which is selecting the most informative feature to be acquired at each point. As it is demonstrated in our experiments, this property is particularly beneficial when a single model is to be used under a budget determined at the prediction-time or any other, not predefined, termination condition.

While it is possible to directly use the measure introduced in (4.7) to find features to be acquired at each time, it would be computationally expensive; because it requires exhaustively measuring the value function for all features at each time. Instead, in addition to a predictor model, we train an action value (i.e., feature value) function which estimates the gain of acquiring each feature based on the current context. For this purpose, we follow the idea of the deep Q-network (DQN) [MKS15, MKS13]. Briefly, DQN suggests end-to-end learning of the action-value function. It is achieved by exploring the space through taking actions using an ϵ-greedy policy, storing experiences in a replay memory, and gradually updating the value function used for exploration. Due to space limitations, readers are referred to [MKS15] for a more detailed discussion.

Figure 4.1 presents the network architecture of the proposed method for prediction and feature acquisition. In this architecture, a predictor network (P-Network) is trained jointly with an action value network (Q-Network). The P-Network is responsible for making prediction and consists of dropout layers that are sampled in order to find the prediction uncertainty. The Q-Network estimates the value of each unknown feature being acquired.

Here, we suggest sharing the representations learned from the P-Network with the Q-Network. Specifically, the activations of each layer in the P-Network serve as input to the adjacent layers of the Q-Network (see Figure 4.1). Note that, in order to increase model stability during the training, we do not allow back-propagation from Q-Network outputs to

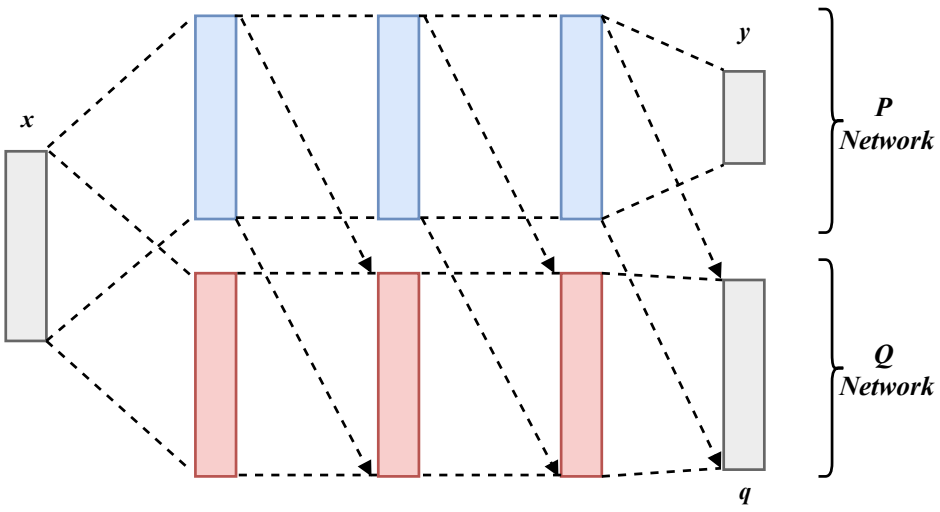


Figure 4.1: Network architecture of the proposed approach for prediction and action value estimation.

P-Network weights. We also explored other architectures and sharing methods including using fully-shared layers between P- and Q-Networks that are trained jointly or only sharing the first few layers. According to our experiments, the suggested sharing method of Figure 4.1 is reasonably efficient, while introducing a minimal burden on the prediction performance.

Algorithm 1 summarizes the procedures for cost-sensitive feature acquisition and training the networks. This algorithm is designed to operate on a stream of input instances, actively acquire features, make predictions, and optimize the models. In this algorithm, if any features are available for free we include them in the initial feature vector; otherwise, we start with all features not being available initially. Here, the feature acquisition is terminated when either a maximum budget is exceeded, a user-defined stopping function decides to stop, or there is no unknown feature left to acquire. It is worth noting that, in Algorithm 1, to simplify the presentation, we assumed that ground-truth labels are available at the beginning of each episode. However, in the actual implementation, we store experiences within an episode in a temporary buffer, excluding the label. At last, after the termination of the feature acquisition procedure, a prediction is being made and upon the availability of label for that sample, the temporary experiences along with the ground-truth label are pushed to the experience replay memory.

In our experiments, for the simplicity of presentation, we assume that all features are

independently acquirable at a certain cost, while in many scenarios, features are bundled and acquired together (e.g., certain clinical measurements). However, it should be noted that the current formulation presented in this chapter allows for having bundled feature sets. In this case, each action would be acquiring each bundle and the reward function is evaluated for the acquisition of the bundle by measuring the variations of uncertainty before and after

acquiring the bundle.

Algorithm 1: Suggested Cost-Sensitive Feature Acquisition, Prediction, and Training

Input: total budget (B), stream of samples (S_i), acquisition cost of features ($\boldsymbol{c}_j$)

Initialize: experience replay memory, random exploration probability (Pr_{rand}) $\leftarrow 1$

for S_i *in the stream* **do**
  $Pr_{rand} \leftarrow decay\_factor \times Pr_{rand}$
  $t \leftarrow 0$, $total\_cost \leftarrow 0$
  $\boldsymbol{x}_i^t \leftarrow$ known features of S_i `// if there are any features available`
  $\tilde{y}_i \leftarrow$ class label of S_i
  $terminate\_flag \leftarrow False$
  while *not terminate_flag* **do** `// collect experiences from each episode`
    if *new random in* $[0, 1) < Pr_{rand}$ **then** `// if it is a random exploration`
      $j \leftarrow$ index of a randomly selected unknown feature
    else `// if it is an exploration using policy`
      $j \leftarrow$ index of the unknown feature with maximum Q value
    $\boldsymbol{x}_i^{t+1} \leftarrow q(\boldsymbol{x}_i^t, j)$ `// acquire feature j`
    $total\_cost \leftarrow total\_cost + \boldsymbol{c_j}$ `// pay the cost of j`
    $r_{i,j}^t \leftarrow \frac{||Cert(\boldsymbol{x}_i^t) - Cert(\boldsymbol{x}_i^{t+1})||}{\boldsymbol{c}_j}$
    push $(\boldsymbol{x}_i^t, \boldsymbol{x}_i^{t+1}, j, r_{i,j}^t, \tilde{y}_i)$ into the replay memory
    $t \leftarrow t + 1$
    if *total_cost* $\geq B$ *or stop_condition() or no unknown feature* **then**
      $terminate\_flag \leftarrow True$ `// terminate if all the budget is used`
    if *update_condition()* **then**
      train_batch $\leftarrow$ random mini-batch from the replay memory
      update P, Q, and target Q networks using train_batch `// Train P & Q`
  end
end

4.2.2 Implementation Details

In this chapter, PyTorch numerical computational library [PGC17] is used for the implementation of the proposed method. The experiments took between a few hours to a couple days on a GPU server, depending on the experiment. Here, we explored fully connected multi-layer neural network architectures; however, the approach taken can be readily applied to other neural network and deep learning architectures. We normalize features prior to our experiments statistically ($\mu = 0, \sigma = 1$) and impute missing features with zeros. Note that, in our implementation, for efficiency reasons, we use NaN (not a number) values to represent features that are not available and impute them with zeros during the forward/backward computation.

Cross-entropy and mean squared error (MSE) loss functions were used as the objective functions for the P and Q networks, respectively. Furthermore, the Adam optimization algorithm [KB14] was used throughout this work for training the networks. We used dropout with the probability of 0.5 for all hidden layers of the P-Network and no dropout for the Q-Network. The target Q-Network was updated softly with the rate of 0.001. We update P, Q, and target Q networks every $1 + \frac{n_{fe}}{100}$ experiences, where n_{fe} is the total number of features in an experiment. In addition, the replay memory size is set to store $1000 \times n_{fe}$ most recent experiences. The random exploration probability is decayed such that eventually it reaches the probability of 0.1. We determined these hyper-parameters using the validation set. Based on our experiments, the suggested solution is not much sensitive to these values and any reasonable setting, given enough training iterations, would result in reasonable a performance. A more detailed explanation of implementation details for each specific experiment is provided in Section 4.3.

4.3 Results and Experiments

4.3.1 Datasets and Experiments

We evaluated the proposed method on three different datasets: MNIST handwritten digits [LCB98], Yahoo Learning to Rank (LTRC) [CC11], and a health informatics dataset. The MNIST dataset is used as it is a widely used benchmark. For this dataset, we assume equal feature acquisition cost of 1 for all features. It is worth noting that we are considering the permutation invariant setup for MNIST where each pixel is a feature discarding the spatial information. Regarding the LTRC dataset, we use feature acquisition costs provided by Yahoo! that corresponding to the computational cost of each feature. Furthermore, we evaluated our method using a real-world health dataset for diabetes classification where feature acquisition costs and budgets are natural and essential to be considered. The national health and nutrition examination survey (NAHNES) data [nha18] was used for this purpose. A feature set including: (*i*) demographic information (age, gender, ethnicity, etc.), (*ii*) lab results (total cholesterol, triglyceride, etc.), (*iii*) examination data (weight, height, etc.) , and (*iv*) questionnaire answers (smoking, alcohol, sleep habits, etc.) is used here. An expert with experience in medical studies is asked to suggest costs for each feature based on the overall financial burden, patient privacy, and patient inconvenience. Finally, the fasting glucose values were used to define three classes: normal, pre-diabetes, and diabetes based on standard threshold values. The final dataset consists of 92062 samples of 45 features.

In the current study, we use reinforcement learning as an optimization algorithm, while processing data in a sequential manner. Throughout all experiments, we used fully-connected neural networks with ReLU non-linearity and dropout applied to hidden layers. We apply MC dropout sampling using 1000 evaluations of the predictor network for confidence measurements and finding the reward values. Meanwhile, 100 evaluations are used for prediction at test-time. We selected these value for our experiments as it showed stable prediction and uncertainty estimates. Each dataset was randomly splitted to 15% for test, 15% for validation, and the rest for train. During the training and validation phase, we use the random exploration mechanism. For comparison of the results with other work in the literature, as

they are all offline methods, the random exploration is not used during the feature acquisition. However, intuitively we believe in datasets with non-stationary distributions, it may be helpful to use random exploration as it helps to capture concept drift. Furthermore, we do model training multiple time for each experiment and average the outcomes. It is also worth noting that, as the proposed method is incremental, we continued feature acquisition until all features were acquired and reported the average accuracy corresponding to each feature acquisition budget.

Table 4.1 presents a summary of datasets and network architectures used throughout the experiments. In this table, we report the number of hidden neurons at each network layer of the P and Q networks. For the Q-Network architecture, the number of neurons in each hidden layer is reported as the number of shared neurons from the P-Network plus the number of neurons specific to the Q-Network.

Table 4.1: The summary of datasets and experimental settings.

Dataset	Instances	Features	Classes	P-Net Architecture	Q-Net Architecture
MNIST [LCB98]	70000	784	10	$[512, 512, 128, 64]$	$[\mathtt{512}+512, \mathtt{512}+256, \mathtt{128}+65, \mathtt{64}+16]$
LTRC [CC11]	34815	519	5	$[128, 32]$	$[\mathtt{128}+128, \mathtt{32}+8]$
Diabetes (Sec. 4.3.1)	92062	45	3	$[64, 32, 16]$	$[\mathtt{64}+64, \mathtt{32}+16, \mathtt{16}+10]$

4.3.2 Performance of the Proposed Approach

Figure 4.2 presents the accuracy versus acquisition cost curve for the MNIST dataset. Here, we compared results of the proposed method (OL) with a feature acquisition method based on recurrent neural networks (RADIN) [CDA16a], a tree-based feature acquisition method (GreedyMiser) [XWC12], and a recent work using reinforcement learning ideas (RL-Based) [JPL17]. As it can be seen from this figure, our cost-sensitive feature acquisition method achieves higher accuracies at a lower cost compared to other competitors. Regarding the RL-Based method, [JPL17], to make a fair comparison, we used the similar network sizes and learning algorithms as with the OL method. Also, it is worth mentioning that the

RL-based curve is the result of training many models with different cost-accuracy trade-off hyper-parameter values, while training the OL model gives us a complete curve. Accordingly, evaluating the method of [JPL17] took more than 10 times compared to OL.

Figure 4.3 presents the accuracy versus acquisition cost curve for the LTRC dataset. As LTRC is a ranking dataset, in order to have a fair comparison with other work in the literature, we have used the normalized discounted cumulative gain (NDCG) [JK02] performance measure. In short, NDCG is the ratio of the discounted relevance achieved using a suggested ranking method to the discounted relevance achieved using the ideal ranking. Inferring from Figure 4.3, the proposed method is able to achieve higher NDCG values using a much lower acquisition budget compared to tree-based approaches in the literature including CSTC [XKW14], Cronus [CXW12], and Early Exit [CZC10].

Figure 4.4a shows a visualization of the OL feature acquisition on the diabetes dataset. In this figure, the y-axis corresponds to 50 random test samples and the x-axis corresponds to each feature. Here, warmer colors represent features that were acquired with more priority and colder colors represent less acquisition priority. It can be observed from this figure that OL acquires features based on the available context rather than having a static feature importance and ordering. It can also be seen that OL gives more priority to less costly and yet informative features such as demographics and examinations. Furthermore, Figure 4.4b demonstrates the accuracy versus acquisition cost for the diabetes classification. As it can be observed from this figure, OL achieves a superior accuracy with a lower acquisition cost compared to other approaches. Here, we used the exhaustive feature query method as suggested by [EFM16] using sensitivity as the utility function, the method suggested by [JPL17] (RL-Based), as well a recent paper using gating functions and adaptively trained random forests [NS17] (Adapt-GBRT).

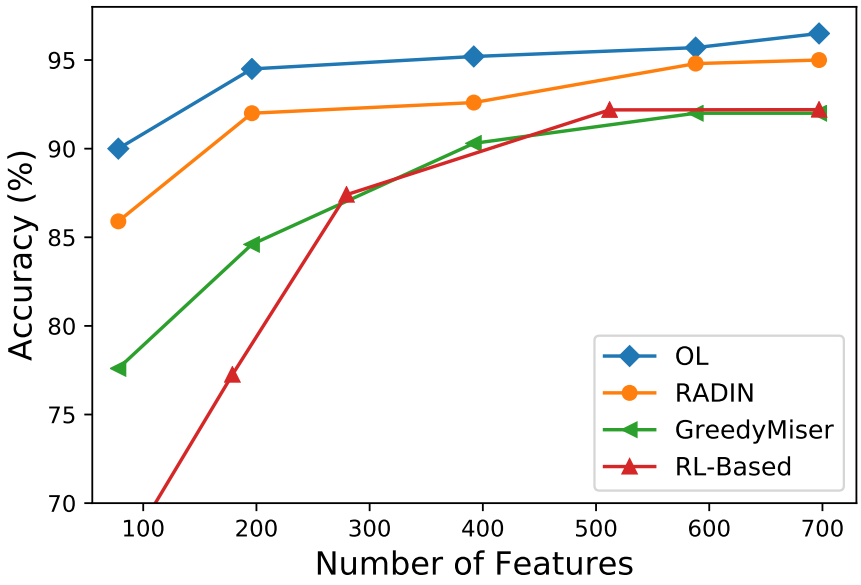


Figure 4.2: Evaluation of the proposed method on MNIST dataset. Accuracy vs. number of acquired features for OL, RADIN [CDA16a], GreedyMiser [XWC12], and a recent work based on reinforcement learning (RL-Based) [JPL17].

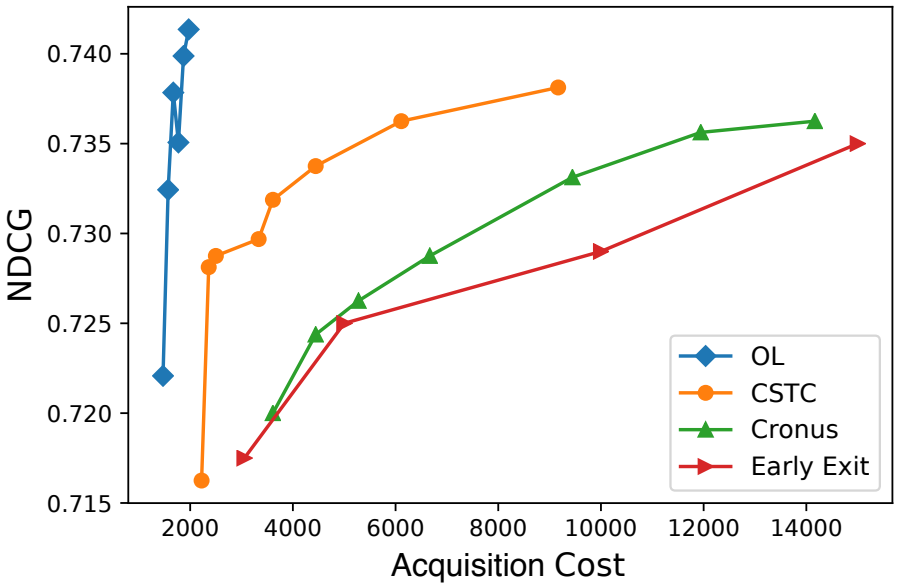


Figure 4.3: Evaluation of the proposed method on LTRC dataset. NDCG vs. cost of acquired features for OL, CSTC [XKW14], Cronus [CXW12], and Early Exit [CZC10] approaches.

4.3.3 Analysis

4.3.3.1 Ablation Study

In this section we show the effectiveness of three ideas suggested in this work i.e, using model uncertainty as a feature-value measure, representation sharing between the P and Q networks, and using MC-dropout as a measure of prediction uncertainty. Additionally, we study the influence of the available budget on the performance of the algorithm. In these experiments, we used the diabetes dataset. A comparison between the suggested feature-value function (OL) in this chapter with a traditional feature-value function (RL-Based) was presented in Figure 4.2 and Figure 4.4b. We implemented the RL-Based method such that it is using a similar architecture and learning algorithm as the OL, while the reward function is simply the the negative of feature costs for acquiring each feature and a positive value for making correct predictions. As it can be seen from the comparison of these approaches, the reward function suggested in this chapter results in a more efficient feature acquisition.

In order to demonstrate the importance of MC-dropout, we measured the average of accuracy at each certainty value. Statistically, confidence values indicate the average accuracy of predictions [GPS17]. For instance, if we measure the certainty of prediction for a group of samples to be 90%, we expect to correctly classify samples of that group 90% of the time. Figure 4.5 shows the average prediction accuracy versus the certainty of samples reported using the MC-dropout method (using 1000 samples) and directly using the softmax output values. As it can be inferred from this figure, MC-dropout estimates are highly accurate, while softmax estimates are mostly over-confident and inaccurate. Note that the accuracy of certainty estimates are crucially important to us as any inaccuracy in these values results in having inaccurate reward values. Figure 4.5b shows the accuracy versus cost curves that the suggested architecture achieves using the accurate MC-dropout certainty estimates and using the inaccurate softmax estimates. It can be seen from this figure that more accurate MC-dropout estimates are essential.

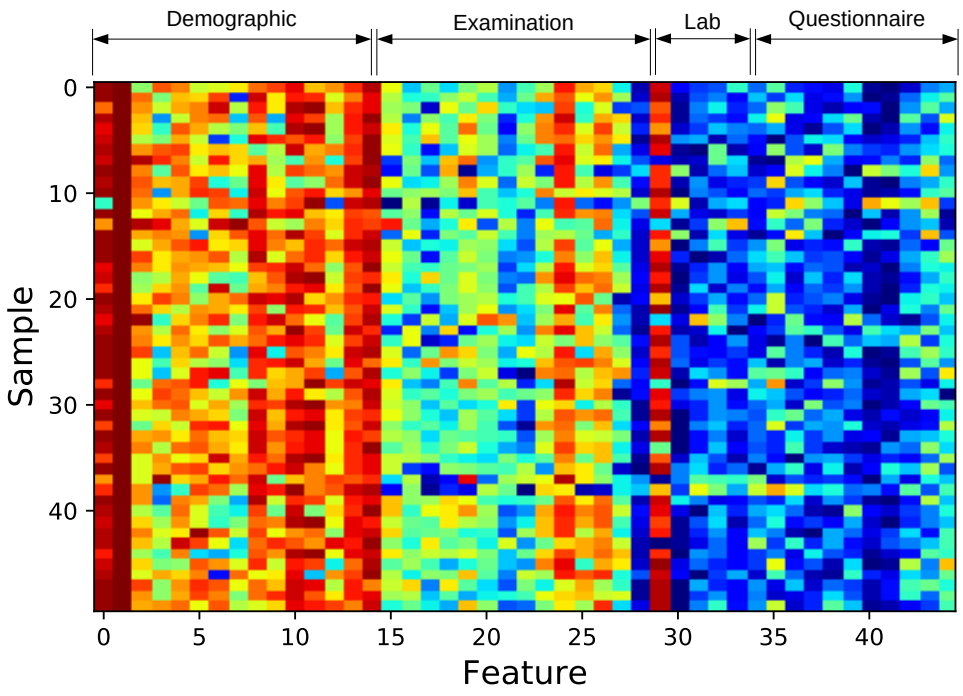
Figure 4.6 demonstrates the speed of convergence using the suggested sharing between the P and Q networks (W/ Sharing) as well as not using the sharing architecture (W/O

Sharing). Here, we use the normalized area under the accuracy-cost curve (AUACC) as measure of acquisition performance at each episode. Please note that we adjust the number of hidden neurons such that the number of Q-Network parameters is the same for each corresponding layer between the two cases. As it can be seen from this figure, the suggested representation sharing between the P and Q networks increases the speed of convergence.

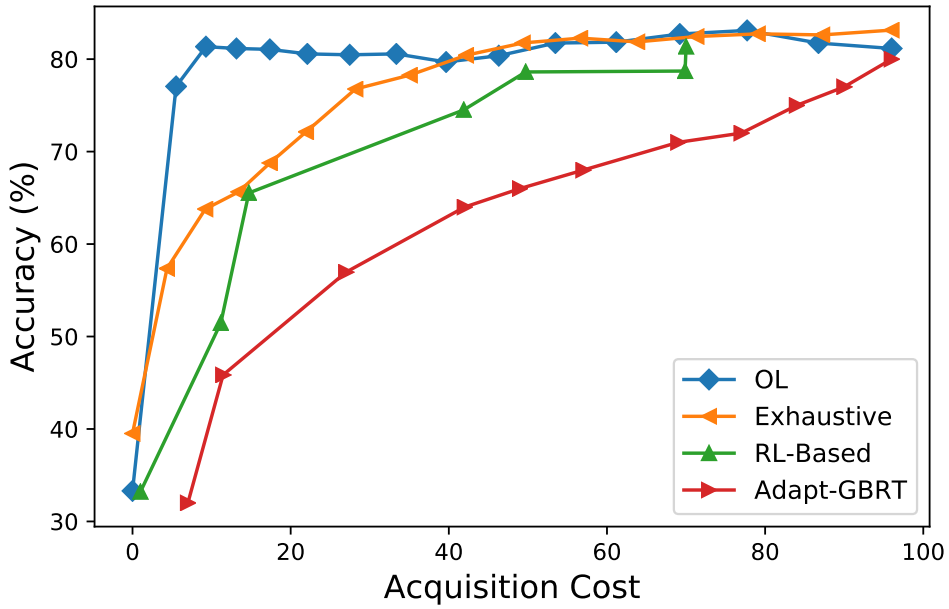
Figure 4.7 shows the performance of the OL method having various limited budgets during the operation. Here, we report the accuracy-cost curves for 25%, 50%, 75%, and 100% of the budget required to acquire all features. As it can be inferred from this figure, the suggested method is able to efficiently operate at different enforced budget constraints.

4.3.3.2 Convergence Analysis

Figure 4.8a and 4.8b demonstrate the validation accuracy and AUACC values measured during the processing of the data stream at each episode for the MNIST and Diabetes datasets, respectively. As it can be seen from this figure, as the algorithm observes more data samples, it achieves higher validation accuracy/AUACC values, and it eventually converges after a certain number of episodes. It should be noted that, in general, convergence in reinforcement learning setups is dependent on the training algorithm and parameters used. For instance, the random exploration strategy, the update condition, and the update strategy for the target Q network would influence the overall time behavior of the algorithm. In this chapter, we use conservative and reasonable strategies as reported in Section 4.2.2 that results in stable results across a wide range of experiments.

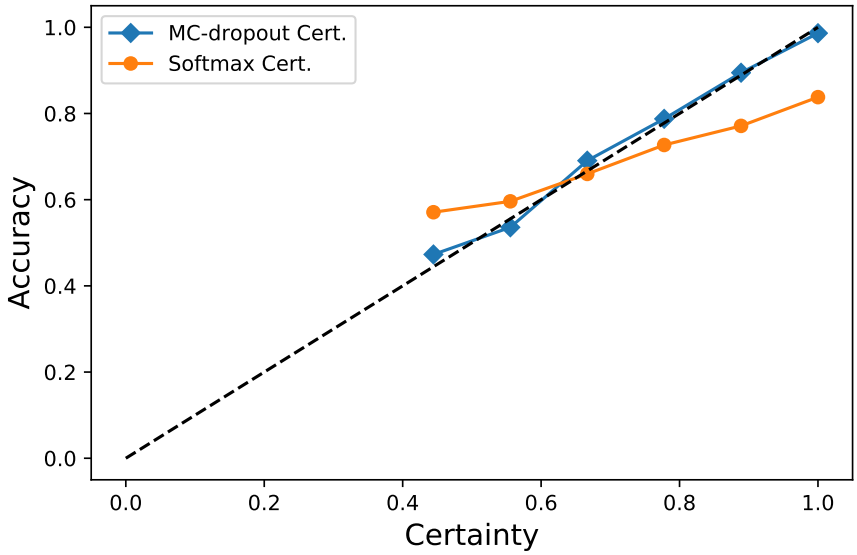


(a)

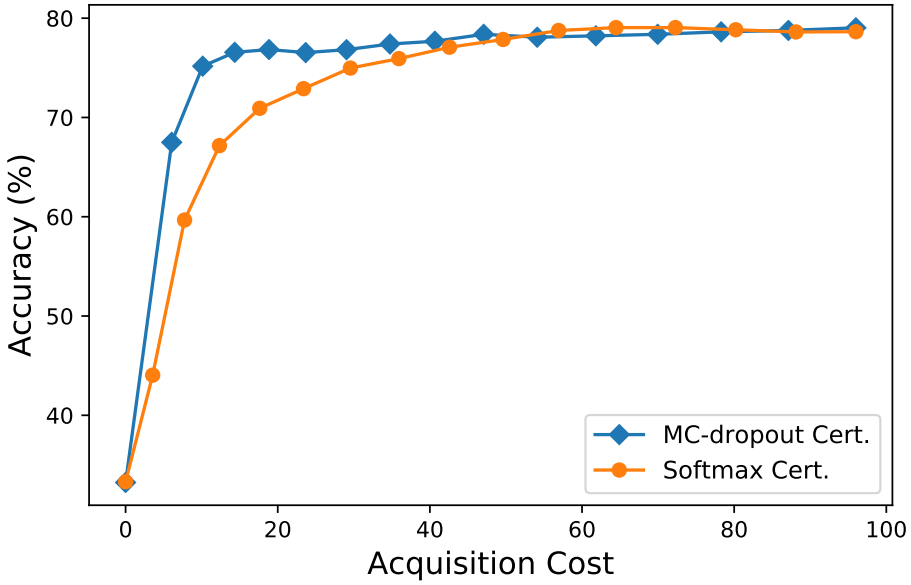


(b)

Figure 4.4: Evaluation of the proposed method on the diabetes dataset. (a) Visualization of feature acquisition orders for 50 test samples (warmer colors represent more priority). (b) Accuracy vs. cost of acquired features for this work (OL), an exhaustive sensitivity-based method (Exhaustive) [EFM16], the method suggested by [JPL17] (RL-Based), and using gating functions and adaptively trained random forests [NS17] (Adapt-GBRT).



(a)



(b)

Figure 4.5: (a) The average prediction accuracy versus the certainty of samples reported using the MC-dropout method (1000 samples) and directly using the softmax output values. (b) The accuracy versus cost curves for using the MC-dropout method and directly using the softmax output values.

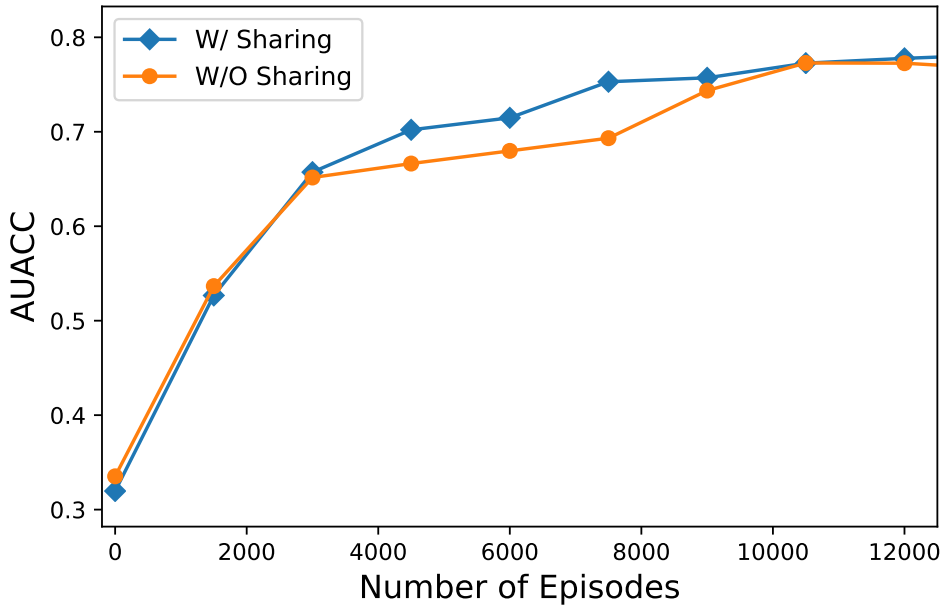


Figure 4.6: The speed of convergence using the suggested sharing between the P and Q networks (W/ Sharing) compared with not using the sharing architecture (W/O Sharing).

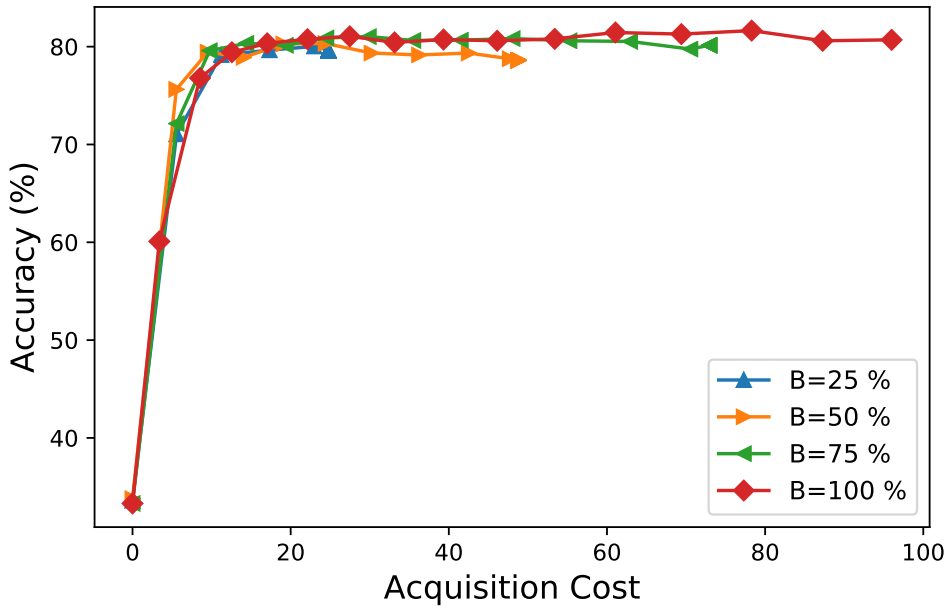
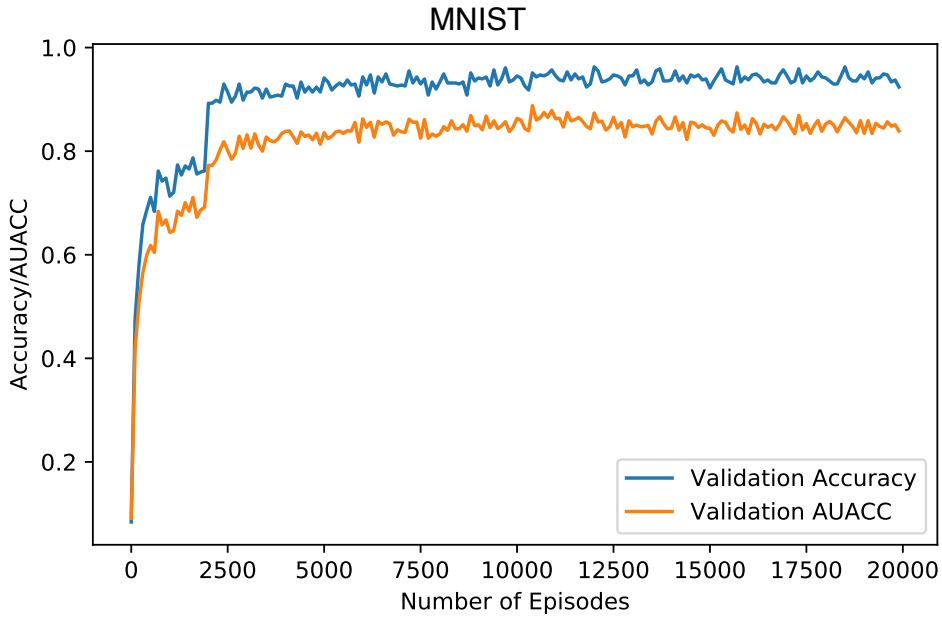
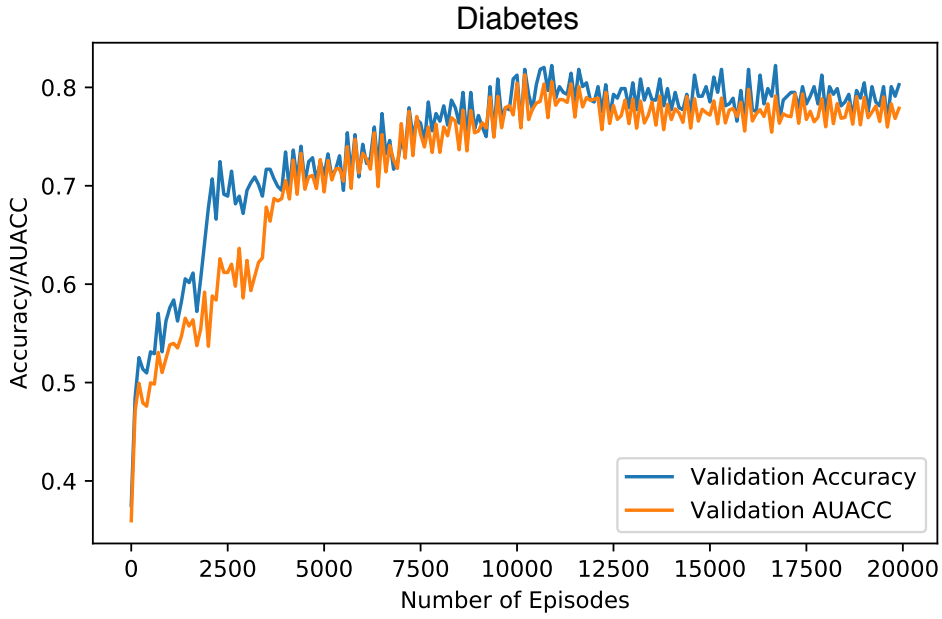


Figure 4.7: Accuracy versus cost curves achieved using different budget levels: 25 %, 50%, 75%, and 100% of the cost of acquiring all features.



(a)



(b)

Figure 4.8: The validation set accuracy and AUACC values versus the number of episodes for the (a) MNIST and (b) Diabetes datasets.

CHAPTER 5

Generative Imputation and Stochastic Prediction[1]

5.1 Problem Definition

We make the general assumption of having access to an incomplete dataset $\mathcal{D}$ consisting of a set of feature vector, mask vector, and target class pairs $(\boldsymbol{x}_i, \boldsymbol{k}_i, y_i)$. For each feature vector, $\boldsymbol{x}_i \in \mathbb{R}^d$, only a subset of the features is available. The mask vector $\boldsymbol{k}_i \in \{0,1\}^d$ is used to indicate available features and missing features by ones and zeros, respectively. Here, to represent features as fixed-width vectors, arbitrary (or NaN) values are used to fill missing values. Also, for convenience, we often use $\boldsymbol{x}_i^{obs}$ and $\boldsymbol{x}_i^{miss}$ to refer to the set of observed and missing features for the feature vector $\boldsymbol{x}_i$. Note that the $(\boldsymbol{x}_i^{obs}, \boldsymbol{x}_i^{miss})$ notation does not use vectors for representation and instead is using sets for a more abstract representation rather than the fixed-length vector notation of $(\boldsymbol{x}_i, \boldsymbol{k}_i)$.

We define our objective in two steps: ***(i)*** Imputing missing values via sampling from the conditional distribution of missing features given observed features i.e., $P(\boldsymbol{x}_i^{miss}|\boldsymbol{x}_i^{obs})$. ***(ii)*** Estimating the distribution of target classes given the observed features and the distribution of missing features i.e., $P(y|\boldsymbol{x}_i^{obs}, \boldsymbol{x}_i^{miss})$. For the first part, we are interested in sampling from the conditional distribution rather than finding the mode of the distribution as the most probable imputation. Similarly, for the second part, we are interested in obtaining a distribution over the possible target assignments and the confidence of each class rather than maximum likelihood class assignments.

---

[1]**This chapter is based on "M. Kachuee, K. Karkkainen, O. Goldstein, S. Darabi, M. Sarrafzadeh, Generative Imputation and Stochastic Prediction, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020." ©2020 IEEE**

5.2 Generative Imputation

To generate samples from the distribution of missing features conditioned on the observed features, we follow the idea first suggested by Yoon *et al.* [YJV18]. In this paradigm, a generator network is responsible for generating imputations while a discriminator is trying to distinguish imputed features from observed features (see Figure 5.1).

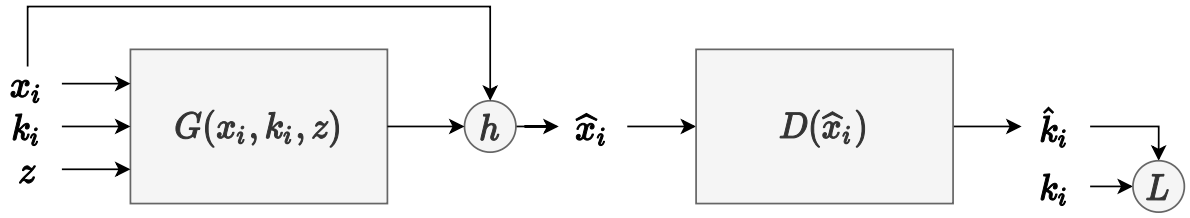


Figure 5.1: Block diagram of the proposed adversarial imputation method. h represents the blending function of (5.1), and L is the adversarial loss function of (5.2).

Specifically, the generator function $G(\boldsymbol{x}_i, \boldsymbol{k}_i, \boldsymbol{z}) \in \mathbb{R}^d$ generates an imputed feature vector, based on observed features, the corresponding mask, and a Gaussian noise vector ($\boldsymbol{z}$). Here, $\boldsymbol{x}_i$ is not revealing any information about missing values as they are represented by invalid values in $\boldsymbol{x}_i$ and are indicated by the mask vector $\boldsymbol{k}_i$. In order to achieve the final imputed vector, $\widehat{\boldsymbol{x}}_i$, we blend (or, merge) the output of the generator with the input features to replace generated values with the exact values of observed features:

$$\widehat{\boldsymbol{x}}_{i,j} = \begin{cases} \boldsymbol{x}_{i,j} & \text{if } \boldsymbol{k}_{i,j} = 1 \\ G(\boldsymbol{x}_i, \boldsymbol{k}_i, \boldsymbol{z})_j & \text{if } \boldsymbol{k}_{i,j} = 0 \end{cases}, \tag{5.1}$$

where $\boldsymbol{x}_{i,j}$ refers to j'th feature of sample i. Also, note that by sampling $\boldsymbol{z}$ multiple times, we can obtain different imputation samples from the conditional distribution indicated by $\widehat{\boldsymbol{x}}_i^l$ where l is the sample number.

A discriminator network, $D(\widehat{\boldsymbol{x}}_i)$, is trained to distinguish real and imputed features by generating a predicted softmax mask output, $\widehat{\boldsymbol{k}}_i$. Here a binary cross-entropy loss per mask element is used as the adversarial objective function:

$$\max_G \min_D L(G, D) = \mathbb{E}_{\boldsymbol{k} \sim \mathcal{D}, \widehat{\boldsymbol{k}} \sim D(G(\boldsymbol{x}, \boldsymbol{k}, \boldsymbol{z}))} \left[\boldsymbol{k}^T \log(\widehat{\boldsymbol{k}}) \; + \; (1 - \boldsymbol{k})^T \log(1 - \widehat{\boldsymbol{k}}) \right]. \tag{5.2}$$

The intuition behind this adversarial loss function is that given a generator function which captures the data distribution successfully, the discriminator would not be able to distinguish the parts of the feature vector that were originally missing.

Compared to Yoon *et al.* [YJV18], the objective function of (5.2) does not have an MSE loss term. Instead, we use recent advances in GAN stabilization and training to improve the training process (see Section 4.2.2). While it is quite prevalent in the adversarial learning literature to use additional loss terms such as mean squared error (MSE) to enhance the quality of generated samples, we decided to keep our solution as simple as possible. Additionally, in our experiments, we provide supporting evidence that this simple loss function enables us to sample from the conditional distribution and prevents biased inclinations toward distribution modes.

5.3 Stochastic Prediction

To capture the distribution of target classes given incomplete data, we suggest the idea of stochastic prediction. As indicated in the previous section, the generator can be used to sample from the conditional distribution. Here, a predictor is trained based on the imputed samples to predict class assignments and to calculate the confidence of these assignments (see Figure 5.2). For instance, for a specific test sample at hand, if a certain missing feature is a strong indicator of the target class, we would like to observe the impact of different imputations for that feature on the final hypothesis.

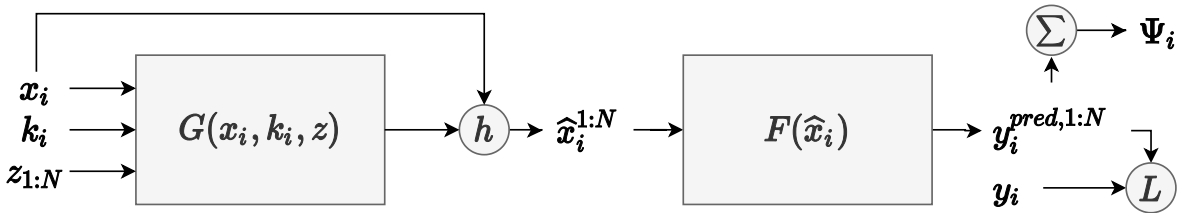


Figure 5.2: Block diagram of the proposed stochastic prediction method. G represents a trained generative imputer (Section 5.2), L is the prediction loss function, and Ψ is the estimated classification certainty defined in (5.7).

Formally, we are interested in finding the certainty of class assignments given observed features:

$$\Psi = P(y|\boldsymbol{x}_i^{obs}). \tag{5.3}$$

Here, Ψ is a vector where each element is representing a certain class. Rewriting (5.3) as a marginal we have:

$$\Psi = \int P(\boldsymbol{x}_i^{miss}) P(y|\boldsymbol{x}_i^{obs}, \boldsymbol{x}_i^{miss})\ d\boldsymbol{x}_i^{miss}. \tag{5.4}$$

Approximating the integration using a summation, given enough samples, Ψ can be estimated by:

$$\Psi \approx \frac{1}{N} \sum P(y|\boldsymbol{x}_i^{obs}, \widehat{\boldsymbol{x}}_i^{miss}), \tag{5.5}$$

where $\widehat{\boldsymbol{x}}_i^{miss}$ are samples taken from the conditional distribution of missing features given observed ones. We use the suggested generative imputation method to generate samples required for this approximation. Rewriting (5.1) using Hadamard product and as function of the noise vector:

$$\widehat{\mathbf{x}}_i = \boldsymbol{k}_i \odot \boldsymbol{x}_i + (1 - \boldsymbol{k}_i) \odot G(\boldsymbol{x}_i, \boldsymbol{k}_i, \boldsymbol{z}) \tag{5.6}$$

Assuming that a predictor, F_θ, is available which predicts class assignments for a complete feature vector, Ψ can be estimated as:

$$\Psi = \mathbb{E}_{\mathbf{z}}[F_\theta(\widehat{\mathbf{x}}_i)] \approx \frac{1}{N} \textstyle\sum_{l=1}^{N} F_\theta(\widehat{\boldsymbol{x}}_i^l)\ . \tag{5.7}$$

Algorithm 2 presents the suggested algorithm for training the predictor. It consists of taking samples from the incomplete dataset, then imputing them using our generator network, and using the imputed samples to update the predictor. Note that, on each epoch and for each sample, the generator generates a new sample from the conditional distribution. Intuitively, it means that the predictor observes and learns to operate under different imputations for a given sample. This is different from approaches such as multiple imputation where several predictors are trained on different imputed versions of a dataset.

Algorithm 3 presents the suggested algorithm for making predictions and estimating target distributions given a trained predictor model. Here, a sample is imputed N times and inference on this set results in an ensemble of predictions over different imputations.

The output of this algorithm can be interpreted as a distribution over the confidence of class assignments given a partially observed test sample. The following claims justify the validity of Algorithm 2 and Algorithm 3.

Algorithm 2: Training the predictor.

Input: G (trained imputer), $\mathcal{D}$ (dataset)

Output: F_θ (trained predictor)

foreach *Training Epoch* **do**
  foreach *($\boldsymbol{x}_i$, $\boldsymbol{k}_i$, y_i) in* $\mathcal{D}$ **do**
    $\boldsymbol{z} \sim N(0, I)$
    $\widehat{\mathbf{x}}_i \leftarrow \boldsymbol{k}_i \odot \boldsymbol{x}_i + (1 - \boldsymbol{k}_i) \odot G(\boldsymbol{x}_i, \boldsymbol{k}_i, \boldsymbol{z})$
    $y_i^{pred} \leftarrow F_\theta(\widehat{\mathbf{x}}_i)$
    $loss \leftarrow L(y_i, y_i^{pred})$
    Backpropagate $loss$
    Update F_θ
  end
end

Claim 1. *(Generalization of the predictor). If we assume imputed $\widehat{x}_i$s are samples from the underlying feature distribution, then the assigned training set labels can be modeled as labels generated from a noisy labeling process.*

Claim 1 permits the analysis of the generalization and convergence for the predictor trained using Algorithm 2 based on current literature in training models with noisy labels [NDR13, RLA14, CLC19]. From the analysis provided by Chen *et al.* [CLC19], test accuracy in asymmetric label noise conditions is a quadratic function of the label noise:

$$P(y_i = \widehat{y}_i) = (1 - \epsilon)^2 + \epsilon^2, \tag{5.8}$$

where $\widehat{y}_i$ is underlying true label for the imputed feature vector ($\widehat{x}_i$), and y_i is the label provided by the incomplete dataset. In (5.8), label noise ratio, ϵ, represents the probability

Algorithm 3: Estimating target distributions.

Input: G (trained imputer), F_θ (trained predictor), $(\boldsymbol{x}, \boldsymbol{k})$ (test sample), N (ensemble samples)

Output: Ψ (distribution over target classes)

$\Psi \leftarrow zeros \in R^{\#classes}$

foreach *Ensemble Sample 1 to N* **do**

  $\boldsymbol{z} \sim N(0, I)$

  $\widehat{\mathbf{x}} \leftarrow \boldsymbol{k} \odot \boldsymbol{x} + (1 - \boldsymbol{k}) \odot G(\boldsymbol{x}, \boldsymbol{k}, \boldsymbol{z})$

  $y^{pred} \leftarrow F_\theta(\widehat{\mathbf{x}})$

  $j \leftarrow argmax(y^{pred})$

  $\Psi_j \leftarrow \Psi_j + \frac{1}{N}$

end

of the label transition from a certain target class to another:

$$\epsilon = 1 - P(\widehat{y_i} = j | y_i = j). \tag{5.9}$$

In practice, ϵ is determined by the problem-specific underlying data distribution as well as the distribution of missing values.

Justification for claim 1 is straightforward, assume that $\{\widehat{y_i}^1 \dots \widehat{y_i}^N\}$ are underlying true labels for each of $\{\widehat{x_i}^1 \dots \widehat{x_i}^N\}$. During training, for any imputed sample in $\{\widehat{x_i}^1 \dots \widehat{x_i}^N\}$, we use the dataset provided label, y_i, to calculate the loss and to update model parameters:

$$Loss_i = \sum_{l=1}^{N} L(y_i, F_\theta(\widehat{x_i}^l)). \tag{5.10}$$

In the case that any of $\{\widehat{y_i}^1 \dots \widehat{y_i}^N\}$ is different from y_i, the loss term corresponding to that term would be calculated using a wrong label. Here, if we consider the average impact on gradients for batches of samples rather than individual cases, the overall impact on training would be very similar to the case of training using noisy labels:

$$Loss = \sum_{i=1}^{|\mathcal{D}|} \sum_{l=1}^{N} L(y_i, F_\theta(\widehat{x_i}^l)), \tag{5.11}$$

where $|\mathcal{D}|$ is the number of dataset samples. Further, in this case, we can find the average label noise as:

$$\epsilon = \frac{\sum_{i=1}^{|\mathcal{D}|} \sum_{l=1}^{N} 1(y_i \neq \widehat{y}_i^{\,l})}{|\mathcal{D}|.N} \tag{5.12}$$

Claim 2. *(Approximation of the target distribution). If we assume:*
(i) imputed $\widehat{x}_i$s are valid samples from the underlying feature distribution: $\widehat{x}_i \sim P(x|x_i^{obs})$,
(ii) a good predictor can be trained using the incomplete data (claim 1),
(iii) enough samples are used and the Monte Carlo estimator is unbiased: $\frac{1}{N}\sum_{l=1}^{N} F_\theta(\widehat{\boldsymbol{x}}_i^l) \rightarrow \mathbb{E}_{\boldsymbol{z}}[F_\theta(\widehat{\boldsymbol{x}}_i)]$ for $N \rightarrow \infty$,
then the target distribution, Ψ, can be estimated accurately.

This claim supports Algorithm 2 that is suggested to estimate the target distribution given a partially observed feature vector.

The first assumption is consistent with the theoretical analysis of generative adversarial networks that they can converge to the true underlying distribution [ARZ18, LBC17]. The second assumption is supported by Claim 1. Regarding the last assumption, each sample requires one forward computation of the generator and predictor networks which, based on the scalability of current network architectures, usually permits thousands of samples to be taken at a reasonable computational cost.

5.4 Implementation Details

As we conduct experiments on image and tabular datasets, we use different architectures for each. For image datasets, we used a generator and discriminator architectures similar to the ones suggested by Wang *et al.* [WLZ18]. However, we improved these architectures using self-attention layers [ZGM18]. It should be noted that, while Zhang *et al.* [ZGM18] suggests using a single self-attention layer in the middle of the network, we observed consistent improvements by inserting multiple self-attention layers before each residual block within the network. Furthermore, as input to the generator, we concatenate input image, mask, and a random $\boldsymbol{z}$ frame along the channels dimension and use it as input. For tabular datasets, we

use a simple 4 layer network consisting of fully-connected and batch-norm layers. Also, the input to the generator is the concatenation of a feature vector, mask vector, and a $\boldsymbol{z}$ vector of size $\frac{1}{8}$ of the input. For all experiments, we use an ensemble size (N) equal to 128.

We used Adam [KB14] for model optimization. Two time-scale update rule (TTUR) [HRU17] was used to balance training the generator and discriminator networks. We explored best TTUR learning-rate settings from the set of {0.001, 0.0005, 0.0001, 0.00005}. Here, Adam parameters β_1 and β_2 are set to 0.5 and 0.999, respectively. Also, spectral normalization was used to stabilize both the generator and discriminator network in our experiments with image data [MKK18]. For the predictor network, we used the default Adam settings as suggested by Kingma *et al.* [KB14]. In all training procedures, we decay learning rate by a factor of 5 after reaching a plateau. For all experiments, we use a batch size of 64. Based on our experiments, we found that pretraining the discriminator while fixing the generator network for the first 5% of the training epochs helps the stability of training.

5.5 Experiments

5.5.1 Datasets

To evaluate the proposed method we use CIFAR-10 [KH09] and MNIST benchmark [LCB98] as image classification datasets as well as five non-image datasets: UCI Landsat [DG17][2], MIT-BIH arrhythmia [MM01], Diabetes, Cholesterol, and Hypertension classification [KKG19] [3] . CIFAR-10 dataset consists of 60,000 32x32 images from 10 different classes. For this task, we use train and test sets as provided by the dataset. As a preprocessing step, we normalize pixel values to the range of [0,1] and subtract the mean image. The only data augmentation we use for this task is to randomly flip training images for each batch.

UCI Landsat consists of 6435 samples of 36 features from 6 different categories. We follow the same train and test split as provided by the dataset. MIT-BIH dataset consists

[2] `https://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)`

[3] `https://github.com/mkachuee/Opportunistic`

of annotated heartbeat signals from which we used the preprocessed version available online[4] consisting of 92,062 samples of 5 different arrhythmia classes. Diabetes dataset is a real-world health dataset of 92,062 samples and 45 features from different categories such as questionnaire, demographics, medical examination, and lab results. The objective is to classify between three different diabetes conditions i.e., normal, pre-diabetes, and diabetes. Similarly, Cholesterol and Hypertension datasets have about 120 features and 50,000 samples each [KKG19]. As MIT-BIH, Diabetes, Cholesterol, and Hypertension datasets do not provide explicit train and test sets, we randomly select 80% of samples as a training set and the rest as a test set. To preprocess our tabular datasets, statistical and unity based normalization are used to balance the variance of different features and center them around zero. Also, while different encoding and representation methods are suggested in the literature to handle categorical features [JGP16, NOG18], here we take the simple approach of encoding categorical variables using one-hot representation and smoothing them by adding Gaussian noise with zero mean and variance equal to 5% of feature variances. In our experiments, we observed a reasonable performance using the suggested simple smoothing; however, more advanced encoding methods are also applicable in this setup and can be applied to enhance the performances even further.

Table 5.1 shows the exact architectures used in this chapter. To show each layer or block we used the following notation. `CxSyPz-t` represents a 2-d convolution layer of kernel size `x`, stride `y`, padding `z`, and number of output channels `t` followed by ReLU activation. `Attn` represents a self-attention layer similar to Zhang *et al.* [ZGM18]. `R-x` represents a residual block consisting of two 2-d convolutions with kernel size 3 (padding size 1), batch normalization, and ReLU activation. `CTxSyPz-t` is the convolution transpose corresponding to `CxSyPz-t`. `FC-x` is representing a linear fully-connected layer of `x` output neurons with biases. We use spectral normalization as suggested by [MKK18] for all convolutional layers in both generator and discriminator networks.

[4] `https://www.kaggle.com/shayanfazeli/heartbeat`

Table 5.1: Network architectures used in our experiments.

Dataset	Generator/Discriminator Architecture	Predictor Architecture
CIFAR-10	C7S1P3-64, C3S2P1-128, Attn, R-128, Attn, R-128, Attn, R-128, Attn, R-128, CT3S2P1-128, CT7S1P3-3, Tanh/Sigmoid	ResNet-18 [HZR16][a]
Landsat	FC-64, Sigmoid, BNorm, FC-64, Sigmoid, BNorm, FC-64, Sigmoid, BNorm, FC-36, Tanh/Sigmoid	FC-64, ReLU, BNorm, FC-64, ReLU, BNorm, FC-6, Softmax
MIT-BIH	FC-1860, ReLU, BNorm, FC-1860, ReLU, BNorm, FC-1860, ReLU, BNorm, FC-186, Tanh/Sigmoid	FC-1860, ReLU, BNorm, FC-1860, ReLU, BNorm, FC-5, Softmax
Diabetes	FC-45, ReLU, BNorm, FC-45, ReLU, BNorm, FC-45, ReLU, BNorm, FC-45, Tanh/Sigmoid	FC-22, ReLU, BNorm, FC-22, ReLU, BNorm, FC-3, Softmax
Cholesterol	FC-242, ReLU, BNorm, FC-242, ReLU, BNorm, FC-242, ReLU, BNorm, FC-121, Tanh/Sigmoid	FC-242, ReLU, BNorm, FC-242, ReLU, BNorm, FC-2, Softmax
Hypertension	FC-240, ReLU, BNorm, FC-240, ReLU, BNorm, FC-240, ReLU, BNorm, FC-120, Tanh/Sigmoid	FC-240, ReLU, BNorm, FC-240, ReLU, BNorm, FC-2, Softmax
MNIST	FC-1568, ReLU, BNorm, FC-1568, ReLU, BNorm, FC-1568, ReLU, BNorm, FC-784, Tanh/Sigmoid	FC-1568, ReLU, BNorm, FC-1568, ReLU, BNorm, FC-10, Softmax

[a] https://github.com/kuangliu/pytorch-cifar

5.5.2 Missingness Mechanisms

In our experiments, we consider MCAR uniform and MCAR rectangular missingness structures. In MCAR uniform, each feature of each sample is missing based on a Bernoulli distribution with a certain missingness probability (i.e., missing rate) independent of other features. In addition to the case of uniform missingness, for image tasks, we use rectangular missingness/observation structure where rectangular regions of dataset images are missing/observed. To control the rate of missingness and decide on the regions that are missing for each case, we use a latent beta distribution that samples rectangular region's width and height such that the average missing rate is maintained. For missing rates less than 50% we make the assumption of having a random rectangular region to be missing, whereas for missing rates more than 50% we assume that only a random rectangular region is observed and the rest of the image is missing.

We would like to note that while the suggested solution in this chapter is readily compatible with MAR structures, in our experiments, to simplify the presentation of results and to have a fair comparison with other work that does not support the MAR assumption, we limited the scope of our experiments to MCAR. Furthermore, to simulate incomplete datasets and to make sure the same features are missing without explicitly storing masks, we use hashed feature vectors to seed random number generators used to sample missing features.

Algorithm 4 presents the procedure used for generating missing values with uniform structure. This algorithm is sampling independent Bernoulli distributions with probabilities equal to the missing rate. Algorithm 5 shows the outline for the rectangular missing structure used in image experiments. It consists of selecting a random point as the center of the rectangle and then deciding on parameters to be used for the beta distribution based on the missing rate. Finally, the width and height of the rectangular region are sampled from the latent beta distribution. In other words, we generate rectangular regions centered at random locations within the image which have width and height values determined by samples from a latent beta distribution. Here, distribution parameters, α and β, are used to control the average missing rate. The outcome would be rectangular regions of different shape at different locations within the frame with the expected portion of missing area equal to the missing rate.

Algorithm 4: MCAR uniform generation.

Input: $\boldsymbol{x}$ (complete feature), r (missing rate)

Output: $\boldsymbol{x_m}$ (incomplete feature)

$seed_x \leftarrow hash(\boldsymbol{x})$

$\boldsymbol{k} \leftarrow 1 - Bernoulli(seed_x, shape(x), prob = r)$

$\boldsymbol{x_m} \leftarrow \boldsymbol{k} \odot \boldsymbol{x} + (1 - \boldsymbol{k}) \odot NaN$

In order to decide on the beta distribution parameters i.e. α and β we use numerical simulations. Specifically, we fix one of the parameters to 1 and change the other parameter in the range of [1,10], while measuring the average missing rate caused by each case. Figure 5.3

Algorithm 5: MCAR rect. generation.

Input: $\boldsymbol{x}$ (complete feature), r (missing rate)

Output: $\boldsymbol{x_m}$ (incomplete feature)

$seed_x \leftarrow hash(\boldsymbol{x})$

$n_x, n_y \leftarrow shape(\boldsymbol{x})$

$(p_x, p_y) \sim (uniform(0, n_x), uniform(0, n_y))$

$\alpha, \beta \leftarrow beta\_params(r)$ `//` $beta\_params$ `gives` α, β `for each missing rate based on numerical simulations`

$(w, h) \sim (Beta(\alpha, \beta) \times n_x), Beta(\alpha, \beta) \times n_y))$

$\boldsymbol{k} \leftarrow rect\_mask(p_x, p_y, w, h)$

$\boldsymbol{x_m} \leftarrow \boldsymbol{k} \odot \boldsymbol{x} + (1 - \boldsymbol{k}) \odot NaN$

shows the missing rates caused by different beta distribution parameters. The first half of Figure 5.3 (missing rates less than about 0.18) corresponds to setting β to 1 and changing α values; and the other half fixing α to 1 and changing β values. To generate missing rates more than 50% we invert our masks and limit the observation to the rectangular region while the rest of the image is missing. Note that missing rates indicate the ratio of features that are missing on the average case. As we are using a latent model for sampling width and height for the rectangles, the actual missing ratios for each specific sample differs between samples. See Table 5.2 for visual examples of different missing rates and missing structures.

5.5.3 Evaluation Measures

Frchet inception distance (FID) [HRU17] score is used to measure the quality of missing data imputation in experiments with images[5]. We also considered using root means squared error (RMSE); however, we decided to only include this result in the appendices as we observed an inconsistent behavior using RMSE in our comparisons as RMSE favors methods that show less variance rather than realistic and sharp samples from the distribution. Also, for each

[5] `https://github.com/mseitzer/pytorch-fid` is adapted to measure the FID scores.

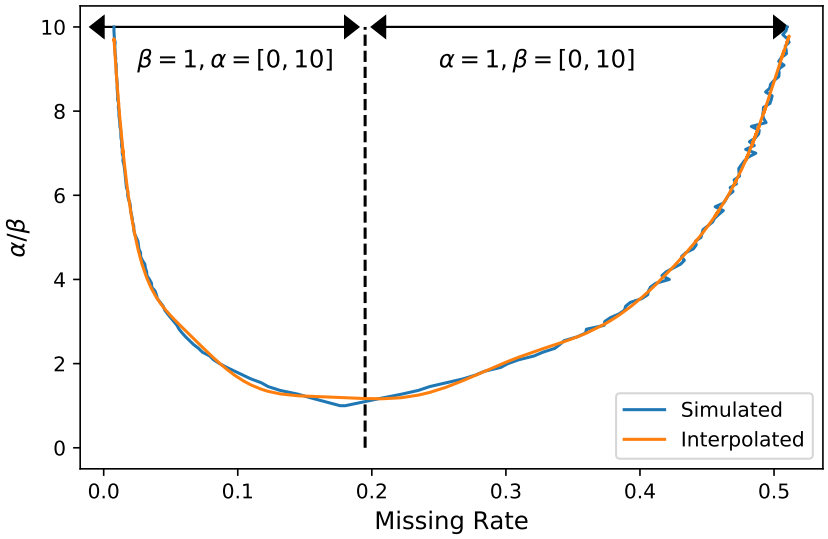


Figure 5.3: Simulation results for measuring average missing rate given different beta distribution parameters.

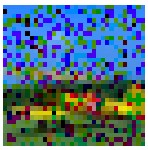
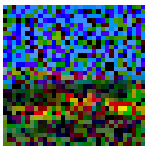
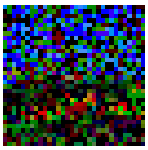
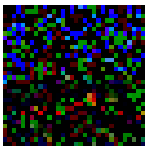
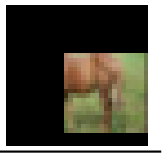
dataset and each missingness scenario, we report top-1 classification accuracy based on the majority vote estimated using Algorithm 3. Another measure that we use is the comparison between the estimated target certainties and average accuracies achieved for each confidence assignment. We run each experiment multiple times: 4 times for CIFAR-10 and 8 times for tabular datasets. We report the mean and standard deviation of results for each case.

We compare our results with MisGAN [LJM19] and GAIN [YJV18] as the state of the art imputation algorithms based on GANs as well as basic denoising autoencoder (DAE) [VLB08] and multiple imputation by chained equations (MICE) [BG10] as baselines. For experiments using MisGAN, we used the same architectures and hyper-parameters as suggested by the MisGAN authors[6]. The only modification was to adapt the last generator layer to generate images with resolutions as we use. Regarding GAIN, we used the same network architecture as our implementation of GI and hyper-parameters as used by the GAIN authors[7]. In the DAE implementation, due to the incomplete data assumption, only observed features appear in the loss function, ignoring reconstruction terms corresponding to missing features. Due

[6] `https://github.com/steveli/misgan`

[7] `https://github.com/jsyoon0823/GAIN`

Table 5.2: Examples of uniform and rectangular missing structures at different missing rates.

	Original	20%	40%	60%	80%
Uniform					
Rect.					

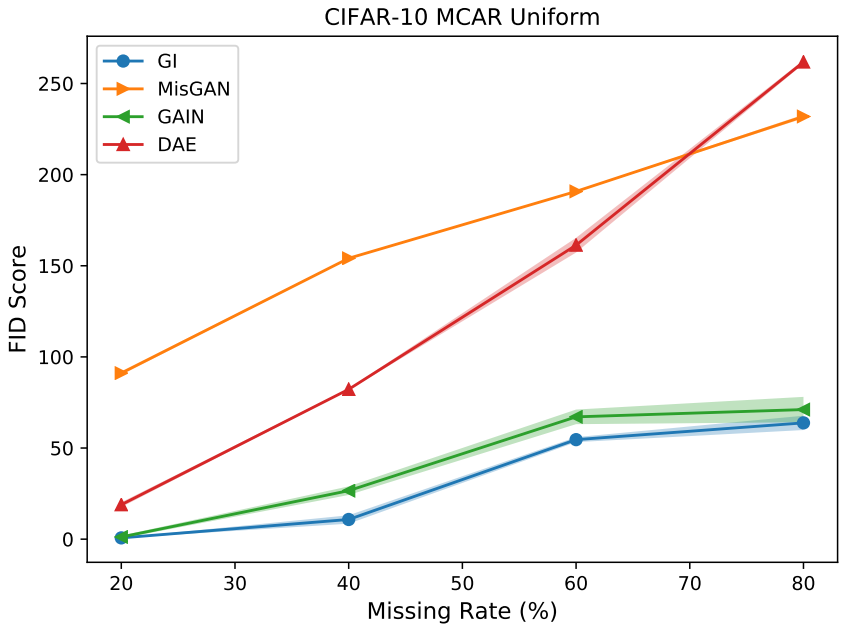
to scalability issues, we were only able to use MICE for the smaller non-image datasets. For these methods, to train and evaluate classifiers, we use predictors trained on imputed datasets rather than the stochastic predictor suggested in Algorithm 2.

5.5.4 Results

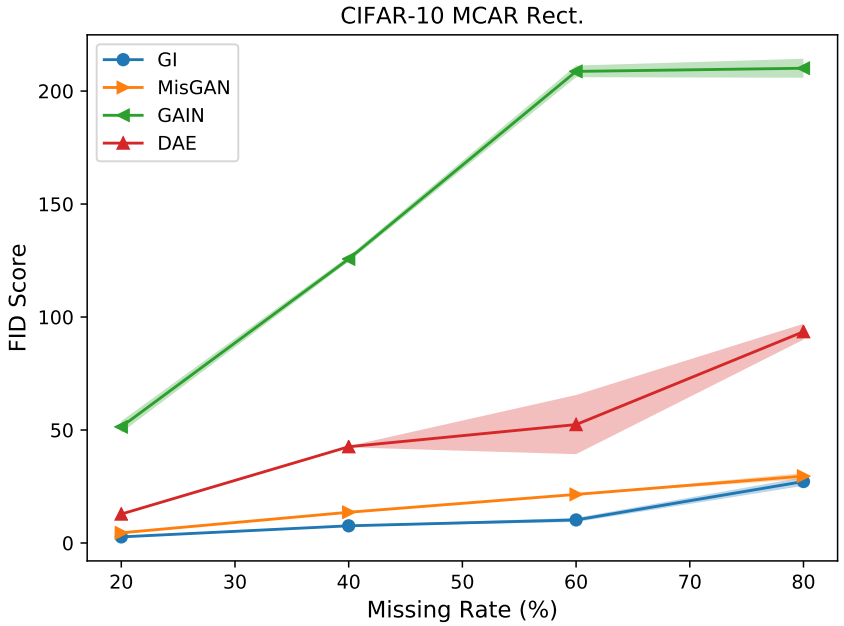
Figure 5.4 presents the comparison of FID scores on the CIFAR-10 dataset at different missing rates for uniform and rectangular missingness. As it can be inferred from these plots, GI outperforms other alternatives in all cases. Also, it can be seen that GAIN is able to provide more reasonable results for uniform missing data structure compared to MisGAN which is mainly effective in the rectangular missing data structure. One possible explanation for this behavior might be the fact that GAIN has an MSE loss term acting similar to an autoencoder loss smoothing noisy missing pixels. On the other hand, MisGAN tries to explicitly model missingness structure and is more successful in capturing a more structured missingness such as the case of a rectangular structure.

Table 5.3 provides a comparison between the top-1 classification accuracy achieved using each method at different missing rates and structures. From this table, GI outperforms other work by achieving the best results in 5 out of 6 cases.

Table 5.4 presents a comparison of classification accuracies for Landsat, MIT-BIH, Di-



(a)



(b)

Figure 5.4: Comparison of FID scores on CIFAR-10 dataset for (a) uniform and (b) rectangular missingness. Lower FID score is better. In many cases, variance values are very small and only observable by magnifying the figures.

Table 5.3: Top-1 CIFAR-10 classification accuracy for different missing rates and structures.

	Accuracy at Missing Rate (%)					
	MCAR Uniform			**MCAR Rect.**		
Method	**20%**	**40%**	**60%**	**20%**	**40%**	**60%**
GI	**89.5** (±0.45)	**87.1** (±0.54)	80.3 (±0.26)	**84.0** (±0.03)	**76.9** (±0.03)	**66.1** (±0.16)
MisGAN	86.5 (±0.31)	83.7 (±0.40)	78.7 (±0.26)	82.9 (±0.44)	75.6 (±0.20)	65.0 (±0.31)
GAIN	88.7 (±0.45)	86.0 (±0.86)	**81.8** (±0.03)	81.7 (±0.03)	73.6 (±0.35)	58.4 (±1.66)
DAE	88.0 (±0.22)	84.0 (±0.50)	79.8 (±0.71)	83.3 (±0.64)	75.5 (±0.44)	63.8 (±0.24)
Mean	85.7 (±0.02)	83.4 (±0.38)	79.2 (±0.16)	82.7 (±0.15)	75.3 (±0.16)	64.0 (±0.32)

abetes, Cholesterol, Hypertension, and MNIST datasets at different missing rates. In the Landsat, Cholesterol, Hypertension, and MNIST benchmarks, GI outperforms other work in all cases. Regarding the MIT-BIH experiemts, GI outperforms other work for missing rates more than 30% while achieving similar accuracies to GAIN for lower missing rates. In the diabetes classification task, GI appears to be most effective imputing missing rates more than 20%.

Figure 5.5 shows a comparison of accuracy versus certainty plots for GI, MisGAN, and GAIN on Landsat dataset at different missing rates. To generate these figures we trained each imputation method and then used Algorithm 2 to train predictors on imputed samples. Finally, Algorithm 3 used to measure the average accuracy at different prediction confidence levels based on a sample of 128 imputations for each test example. As it can be seen from the plots, GI provides results closest to the ideal case of having average confidence values equal to average accuracies. As suggested in (5.7) and supported by the experimental results, the proposed method is better calibrated compared to the traditional approach of imputing each sample as a preprocessing step prior to the prediction, ignoring the imputation uncertainties.

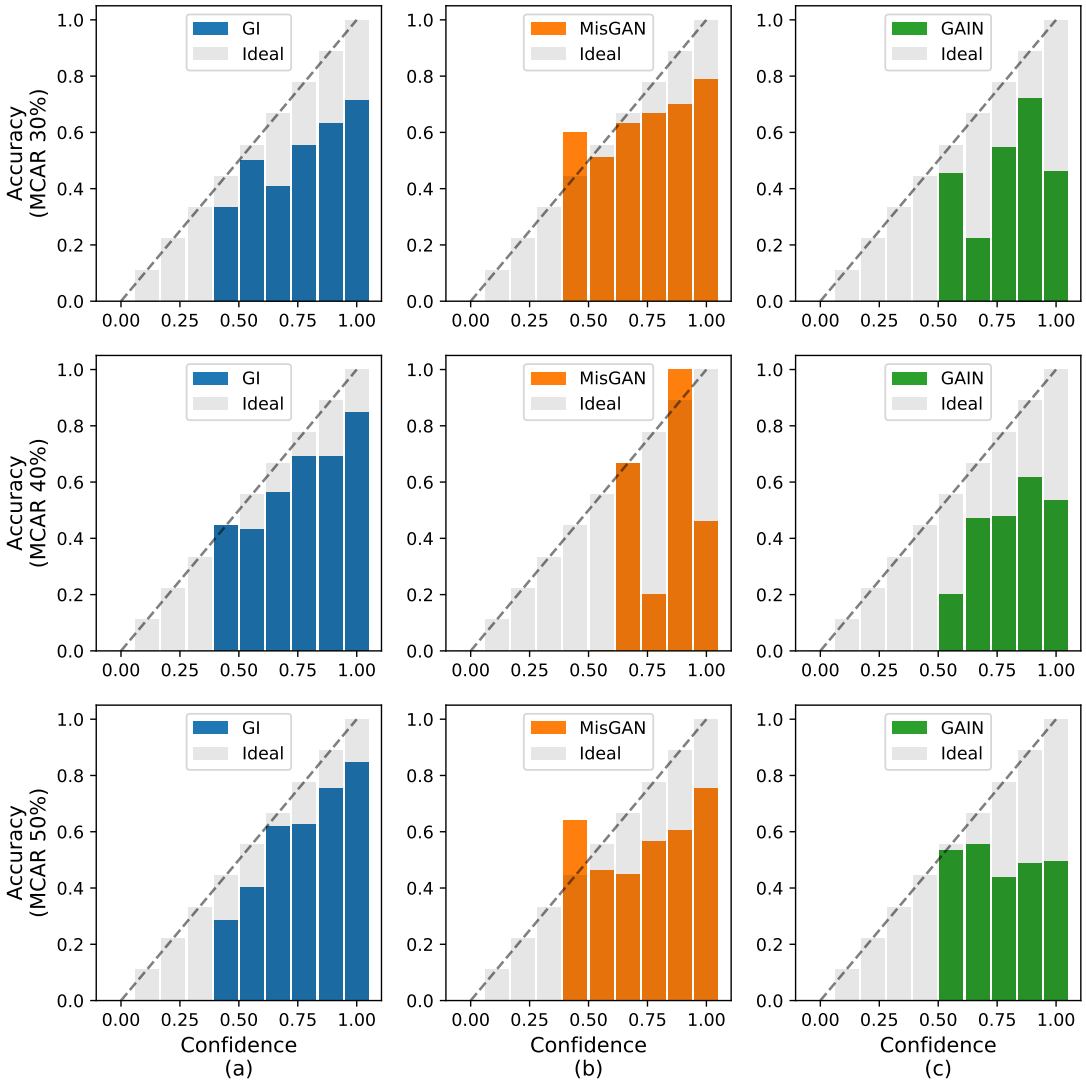


Figure 5.5: Accuracy versus certainty plots for (a) GI, (b) MisGAN, and (c) GAIN on Landsat dataset at the missing rate of 30%, 40%, and 50%.

5.5.5 Visualization using Synthesized Data

In order to provide further insight into the operation of GI and how imputations can potentially influence the outcomes of predictions, we conduct experiments on a synthesized dataset. The original underlying data distribution is generated by sampling 5000 samples from 4 Gaussians of standard deviation 0.1 centered on the vertices of a unit square. We assign two different classes to each cluster such that diagonal vertices are of the same class

(see Figure 5.6a, classes are represented with colors). From this underlying distribution, we make an incomplete dataset with 50% of values missing.

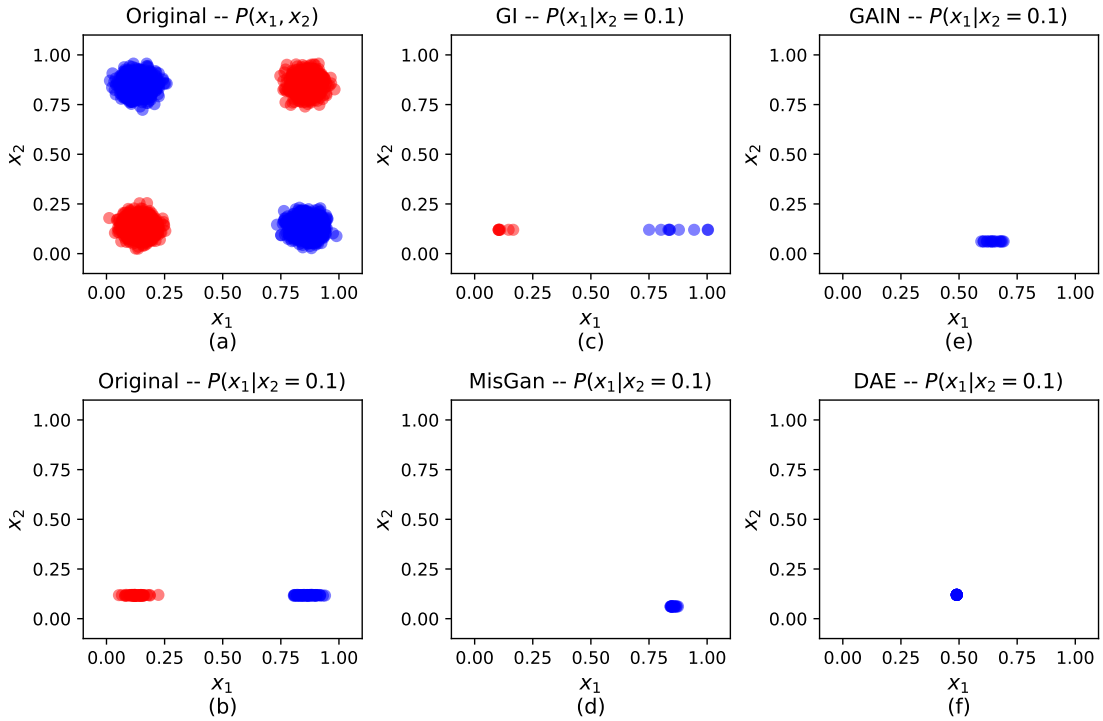


Figure 5.6: Evaluation using synthesized data: (a) samples from the underlying distribution, (b) samples from the conditional underlying distribution, (c-f) samples from the conditional distribution generate by GI, MisGAN, GAIN, and DAE.

The incomplete synthesized dataset is used to train GI and other imputation methods. We take a random test sample in which the second feature has a value of about 0.1 and the other feature is missing. Ideally, in the imputation phase, we would like to sample from the condition distribution i.e. $P(x_1|x_2 = 0.1)$ (see Figure 5.6b). Here, in the prediction phase, an ideal method would decide on not making a confident classification and report the uncertainty. Note that solely observing the value of 0.1 for the second feature does not provide any useful evidence for the prediction. Figure 5.6c-f provide samples and classification results for GI, MisGAN, GAIN, and DAE. As it can be inferred from these figures, GI generates samples relatively similar to samples from the conditional distribution, and it also reflects this uncertainty over the prediction. On the other hand MisGAN, probably due to

its complexity of using three different generators and discriminator pairs, is suffering from mode collapse to a higher degree and is unable to generate samples from the other class, resulting in over-confident assignments. Note that mode collapse is a well-known shortcoming of GANs, and although we observed a better behavior in our models, the results do not perfectly match the ground-truth distribution. GAIN, perhaps due to the MSE loss terms, is inclined towards the mean of the conditional distribution at the origin. DAE, as expected, due to its MSE loss term, only captures the expected value of the distribution mean hence reducing the MSE error and generates over-smoothed imputations.

5.6 Ablation Study

5.6.1 Impact of the Self-Attention Layers

Figure 5.7 presents a comparison between using (GI W/ Atten.) and not using (GI W/O Atten.) self-attention layers before each residual block in the proposed architecture. We report FID scores on CIFAR-10 with rectangular missingness. As it can be inferred from this comparison, using self-attention achieves a consistent improvement over the baseline. We also examined the case of uniform missingness; however, we did not observe any significant improvement for this case. One possible explanation could be the fact that imputing missing data with a uniform structure can be done by processing local regions and does not require attending to different distant regions across the image.

5.6.2 Impact of Ensemble Size

Figure 5.8 shows a comparison of classification accuracies for the Landsat dataset achieved using different ensemble sizes (N). As it can be seen from this figure, higher values of N result in improved accuracies, especially for higher missing rates. Also, it can be observed that for N values more than 64 the difference is negligible.

To study the benefits of the suggested stochastic predictor, we conducted experiments comparing GI with its non-stochastic variation (N=1). Here, the CIFAR-10 dataset with

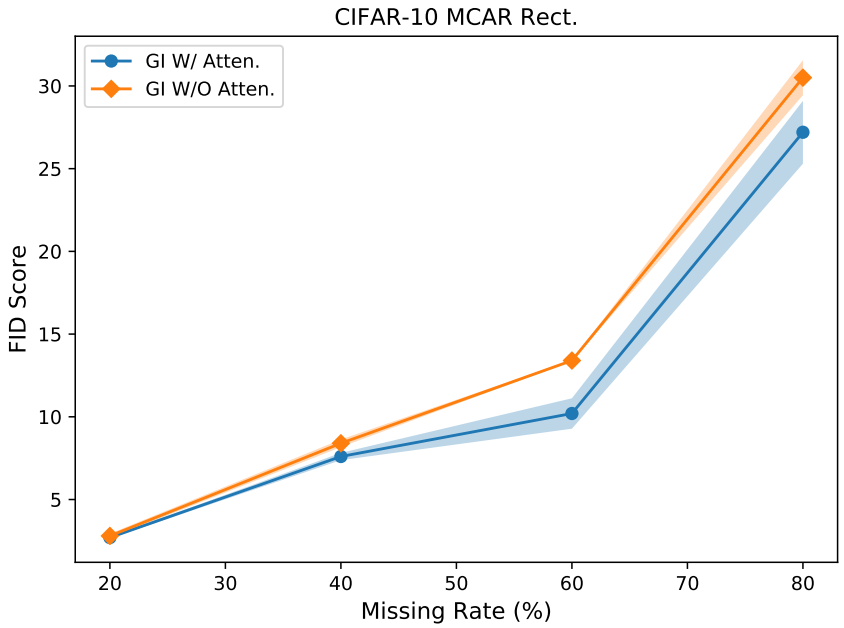


Figure 5.7: Comparison of FID scores achieved with (GI W/ Atten.) and without (GI W/O Atten.) self-attention layers on CIFAR-10 dataset and rectangular missingness. Lower FID score is better.

the rectangular missing structure and missing rates from 20% to as high as 90% is used. From Table 5.5 it can be inferred that as the rate of missingness increases, the benefits of the suggested predictor algorithm increase significantly. We hypothesize that at higher rates of missingness, the conditional distribution of missing features becomes multimodal. In such a scenario, the suggested method captures the uncertainties over the target distribution resulting in the predictor to make more reliable class assignments.

5.6.3 Impact of Training Noise

Addition of noise to input vectors often serves as an input augmentation and results in improved generalization accuracies. In order to verify that the improved GI performance is not merely due to the introduction of noise in the suggested architecture, we conducted an experiment by adding different amounts of Gaussian noise during the training process for GAIN and GI. Specifically, we compared how the CIFAR-10 test accuracies change at

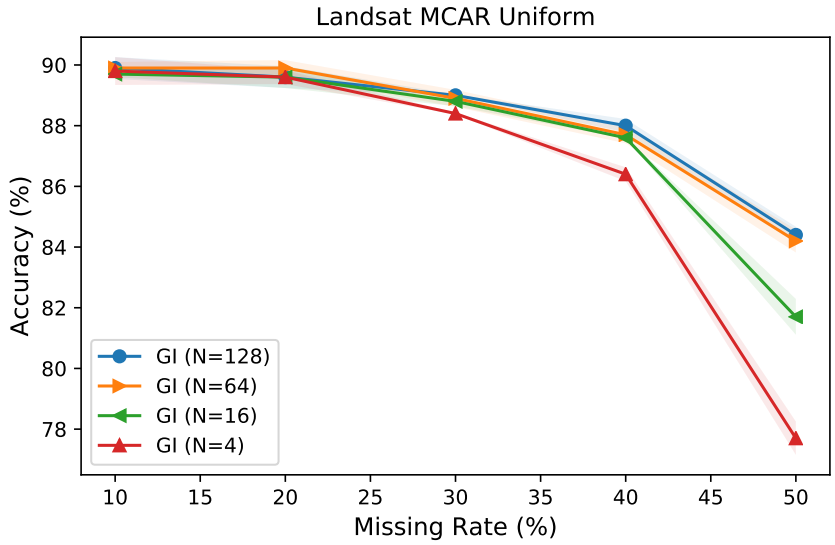


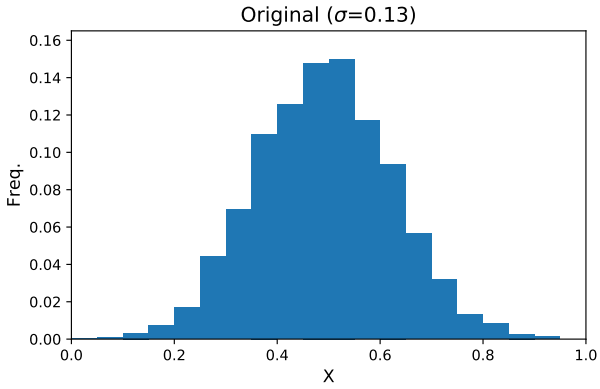
Figure 5.8: Comparison of classification accuracies achieved with different ensemble size (N).

different degrees of training noise for uniform and rectangular missingess structures at the average missing rate of 40%.

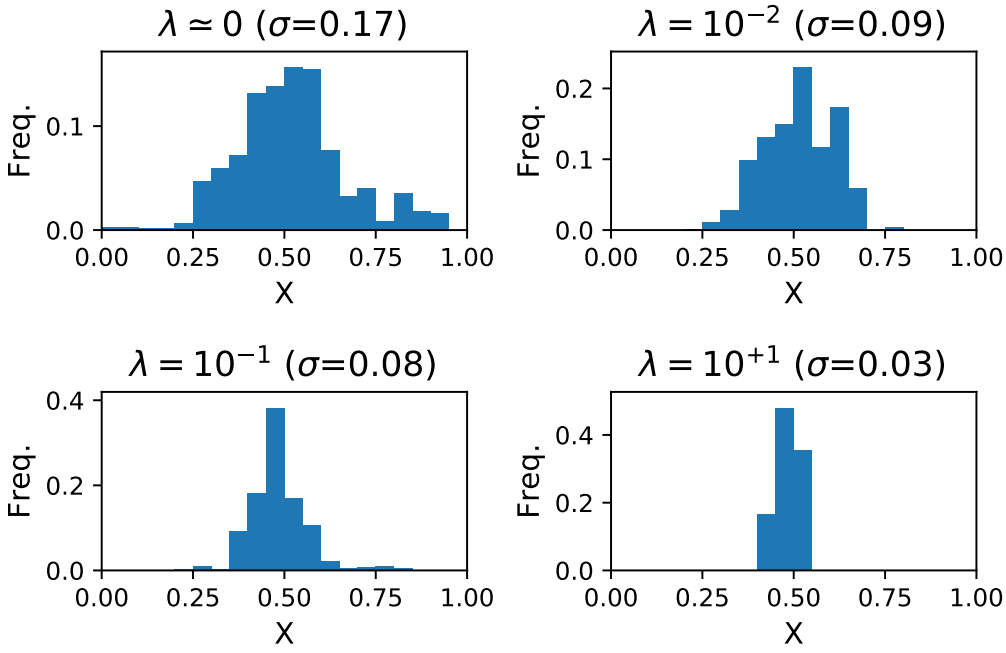
According to Table 5.6, adding small amounts of Gaussian noise (e.g., std=0.0125) improves the generalization under uniform missingness for both GI and GAIN. Even in this case, GI is still outperforming GAIN in terms of final classification performance. It is also interesting to point out that for the case of rectangular missingness adding Gaussian noise results in a consistent reduction in the classification accuracy for both methods.

5.6.4 Impact of the MSE Loss Term

In our earlier discussions, we stated that the MSE loss term used in GAIN would bias the distribution of generated samples toward the mean of the distribution. Here, a synthesized dataset is used to illustrate the impact of MSE loss term on the distribution of generated samples. A hyperparameter, λ, controls the weight of the MSE term in the final objective function. As it can be observed from Figure 5.9, the higher the λ parameter, the lower the variance of the generated samples (i.e., more bias toward the mean of the distribution).



(a)



(b)

Figure 5.9: Comparison of generating samples from a Gaussian distribution (a) samples from the original distribution, (b) samples generated using GAIN imputers with different significance of the MSE term (controlled by λ).

5.6.5 Impact of the Discriminator Hint Vector

Yoon *et al.* [YJV18] suggested the idea of guiding the discriminator network using a hint mechanism. A hint vector reveals a subset of features that are missing to the discriminator. In Figure 5.10 and 5.11 we provide a comparison of learning curves for GI implemented using different hint rates. From Figure 5.10, using the hint mechanism does not result in any noticeable improvement in the final imputation quality justifying the added complexity. For the case of the rectangular missing structure in Figure 5.11; however, using the hint vector causes instabilities in the training process. One possible explanation is: providing even a small portion of the mask as a hint, due to the deterministic nature of the rectangular shape it is equivalent to providing region boundaries to the discriminator making it obvious for the discriminator. In GAN training we generally want to have equal competition between the generator and discriminator.

5.6.6 Visual Results

Figure 5.12 provides examples of masked CIFAR-10 images that are imputed using the proposed method. The variance among imputed samples is representing different possibilities for completing the missing parts. For each input sample, we also show the class assignment certainties estimated from an ensemble of 128 imputations, of which three randomly selected samples are shown here. In certain examples, the missing part is not causing a noticeable uncertainty over target assignments, while in others it leads to some confusion over target assignments based on the different viable imputations.

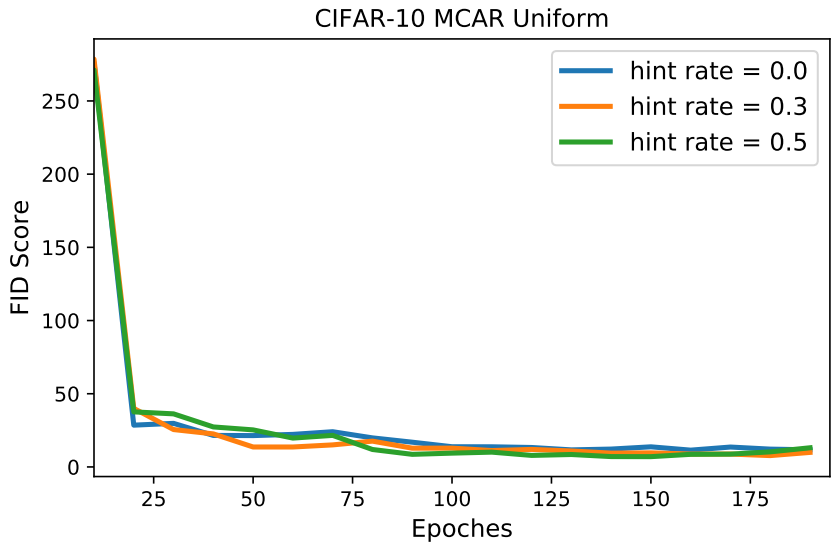


Figure 5.10: Learning curves for CIFAR-10 with uniform missing structure at different discriminator hint rates.

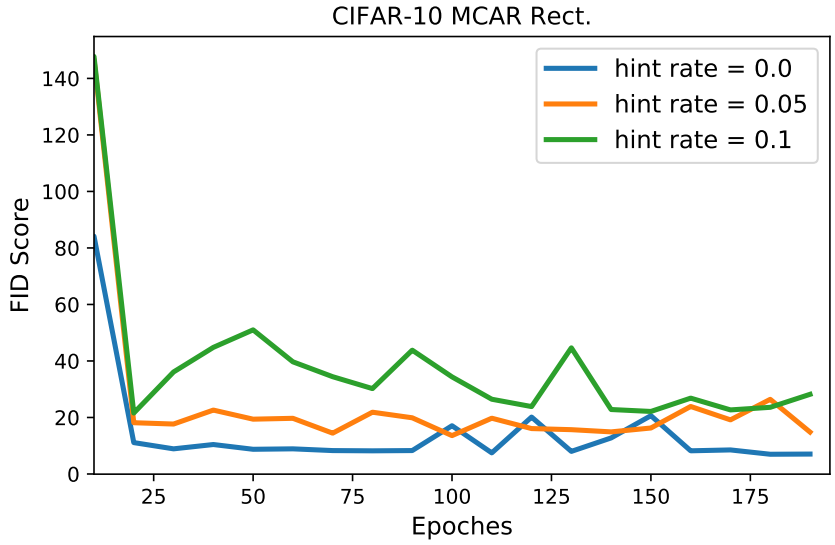


Figure 5.11: Learning curves for CIFAR-10 with rectangular missing structure at different discriminator hint rates.

Table 5.4: Comparison of classification accuracies at different missing rates.

		Accuracy at Missing Rate (%)[a]			
Dataset	**Method**	**10%**	**20%**	**30%**	**40%**
	GI	**89.9** (±0.36)	**89.6** (±0.36)	**89.0** (±0.03)	**88.0** (±0.22)
	MisGAN	87.2 (±0.01)	85.7 (±0.19)	84.0 (±0.61)	82.9 (±0.75)
Landsat [DG17]	GAIN	89.7 (±0.42)	89.4 (±0.56)	88.4 (±0.71)	87.7 (±0.10)
	DAE	89.4 (±0.10)	88.6 (±0.54)	87.5 (±0.14)	86.6 (±0.21)
	MICE	89.5 (±0.16)	89.3 (±0.10)	88.1 (±0.49)	87.5 (±0.03)
	GI	**98.5** (±0.02)	**98.4** (±0.03)	**98.2** (±0.07)	**97.7** (±0.03)
	MisGAN	97.8 (±0.13)	97.4 (±0.07)	96.7 (±0.07)	96.2 (±0.09)
MIT-BIH [MM01]	GAIN	**98.5** (±0.02)	**98.4** (±0.06)	98.0 (±0.09)	97.5 (±0.18)
	DAE	98.4 (±0.02)	98.2 (±0.11)	97.9 (±0.09)	97.4 (±0.02)
	MICE	98.4 (±0.01)	98.3 (±0.01)	98.1 (±0.01)	97.5 (±0.12)
	GI	89.6 (±0.13)	**89.0** (±0.03)	**88.2** (±0.62)	**86.8** (±0.38)
	MisGAN	89.7 (±0.01)	88.9 (±0.30)	87.6 (±0.02)	86.4 (±0.68)
Diabetes [KKG19]	GAIN	89.2 (±0.09)	88.3 (±0.02)	86.9 (±0.09)	83.8 (±1.44)
	DAE	89.3 (±0.05)	88.2 (±0.19)	86.9 (±0.09)	84.8 (±0.03)
	MICE	**89.8** (±0.08)	88.8 (±0.01)	88.0 (±0.08)	86.1 (±0.02)
	GI	**73.2** (±0.12)	**72.2** (±0.14)	**71.6** (±0.30)	**70.4** (±0.20)
	MisGAN	72.8 (±0.31)	71.6 (±0.13)	70.7 (±0.15)	69.9 (±0.13)
Cholesterol [KKG19]	GAIN	72.8 (±0.22)	71.7 (±0.27)	71.2 (±0.05)	70.1 (±0.08)
	DAE	73.0 (±0.19)	71.6 (±0.24)	70.8 (±0.33)	70.2 (±0.04)
	MICE	71.2 (±0.10)	69.9 (±0.13)	68.7 (±0.02)	67.3 (±0.21)
	GI	**77.8** (±0.15)	**77.3** (±0.32)	**77.2** (±0.30)	**76.2** (±0.07)
	MisGAN	77.0 (±0.21)	76.9 (±0.16)	76.1 (±0.04)	75.4 (±0.49)
Hypertension [KKG19]	GAIN	77.5 (±0.10)	76.8 (±0.25)	76.6 (±0.37)	75.8 (±0.14)
	DAE	77.5 (±0.30)	76.8 (±0.11)	76.4 (±0.58)	76.1 (±0.05)
	MICE	76.7 (±0.19)	75.9 (±0.08)	74.7 (±0.20)	73.0 (±0.16)
	GI	**99.0** (±0.01)	**98.9** (±0.07)	**98.7** (±0.01)	**98.6** (±0.01)
	MisGAN	98.6 (±0.02)	98.3 (±0.06)	98.1 (±0.02)	97.5 (±0.06)
MNIST [LCB98]	GAIN	98.8 (±0.02)	98.7 (±0.03)	98.6 (±0.02)	98.5 (±0.03)
	DAE	98.8 (±0.08)	98.7 (±0.03)	98.5 (±0.08)	98.0 (±0.06)
	MICE	98.7 (±0.02)	98.6 (±0.06)	98.4 (±0.07)	98.3 (±0.06)

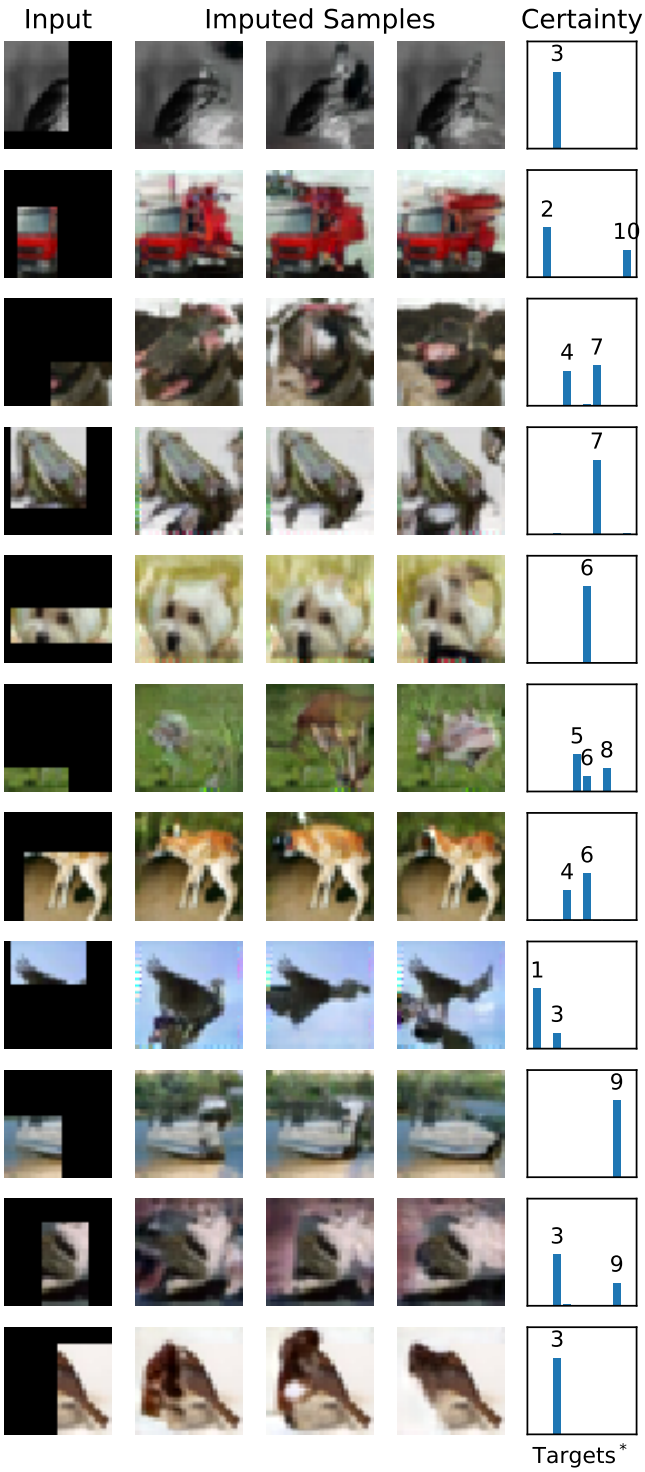
[a]Baseline accuracies for complete datasets (zero missing rate) are Landsat:90.9%, MIT-BIH:98.6%, Diabetes:90.7%, Cholesterol:73.6%, Hypertension:77.9%, MNIST:99.2%.

Table 5.5: Comparison of CIFAR-10 accuracies for the stochastic (N=128) and the deterministic (N=1) predictor under rectangular missingness.

	Accuracy at Missing Rate (%)					
Method	**20%**	**40%**	**60%**	**70%**	**80%**	**90%**
GI (N=128)	84.0	76.9	66.1	59.1	46.0	32.1
GI (N=1)	83.6	75.7	65.1	56.7	42.8	29.4
% difference (normalized)	0.5	1.6	1.5	4.1	6.9	8.4

Table 5.6: Top-1 CIFAR-10 classification accuracy at 40% missing rate using added training noise.

	Accuracy (%)			
	MCAR Uniform (40%)		**MCAR Rect. (40%)**	
Noise STD	**GI**	**GAIN**	**GI**	**GAIN**
0.0	87.1	86.0	**76.9**	73.6
0.0125	**87.3**	86.3	76.8	73.3
0.025	86.5	86.6	76.7	73.2
0.05	85.6	84.7	73.7	72.4
0.1	82.0	80.6	68.7	67.0



*1:plane, 2:car, 3:bird, 4:cat, 5:deer, 6:dog, 7:frog, 8:horse, 9:ship, 10:truck,

Figure 5.12: Visual samples of imputed CIFAR-10 images and estimated classification certainties from each incomplete input. The estimated certainty for each target class assignment is represented by a bar for each class, where the height shows the relative confidence.

CHAPTER 6

Conclusion

In this thesis, we investigated algorithms and methods for cost-sensitive and context-aware learning. Specifically, we suggested a novel approach based on sensitivity analysis in autoencoder architectures. Alternatively, a deep reinforcement learning solution is proposed to enable online cost-sensitive learning from data streams. We evaluated the suggested methods on multiple real-world and synthesized datasets. According to the experimental results, the proposed methods are capable of efficiently acquiring features while accounting for the feature costs and context. Additionally, as acquiring only a subset of features is synonymous with having missing values, proper handling of missing values is a critical component in any real-world realization of cost-sensitives learning systems. We hypothesize that having missing values entails an uncertainty over the value of each missing feature which should be appropriately modeled and considered in the inference. As a part of this thesis, we address this problem by suggesting a method to impute missing values and to estimate prediction uncertainties. We conduct extensive experiments and ablation studies to demonstrate the importance of proper handling of missing values and the effectiveness of the suggested method.

APPENDIX A

Nutrition and Health Data for Cost-Sensitive Learning

[1]

A.1 Methodology

A.1.1 Data Source

We use the National Health and Nutrition Examination Survey (NHANES) [nha18] between 1999 to 2016 as our data source. NHANES is an ongoing survey which is designed to assess the well-being of adults and children in the United States. Each year, health and nutrition data is collected from few thousand individuals and consisting of demographics, questionnaire, examination, and laboratory data. Not all data is collected from each individual (e.g., certain blood tests are not used for young children) and there is a slight variation between the information being collected in each year (e.g., the prevalence of disease change over time causing changes on the data collection focus). For more information about NHANES please refer to the documentation[nha18],[2].

A.1.2 Data Preparation

We developed a general data processing pipeline which can be used for different tasks and settings. The data preparation starts with loading raw data files associated with each variable in the dataset containing values of that variable for each subject. Here, a variable can be

[1]**This chapter is based on: "M. Kachuee, K. Karkkainen, O. Goldstein, D. Zamanzadeh, M. Sarrafzadeh, Cost-Sensitive Diagnosis and Learning Leveraging Public Health Data, *arXiv:1902.07102*, 2019."**

[2]`https://www.cdc.gov/nchs/nhanes`

answer to a demographics question, a certain factor in a blood test, result of a certain examination, etc. Please note that we merge columns and rows based on variable and subject identifiers so that, logically, all variables appear as columns and individuals appears as a row. For a certain task, any available variable could be defined as a feature or a target, depending on the task.

The dataset consists of 9385 unique variables of different types including categorical, real-valued, multiple choice, etc. Accordingly, we use different preprocessing functions such as statistical normalization for real-valued variables and one-hot encoding for categorical variables to prepare each variable for further analysis. Also, it should be noted that, for each individual, only a subset of these variables is available and the rest are missing.

To define each certain task (e.g. diabetes classification), a target should be defined as a function of available variables (e.g., blood glucose level categories). Two methods are suggested to determine variables to be used as input features and appropriate preprocessing functions : (i) explicitly defining a list of variables and their corresponding preprocessing functions (ii) automatically selecting relevant features by searching over the space of variables and automatically deciding on appropriate preprocessing functions. Regarding the second method, one can limit the number of features being eventually used by setting a threshold on the mutual information between the target and each feature as well as a threshold on the percentage of available (not missing) features for each selected variable. Applying this thresholds limits the feature set to features that are informative for the task and are available for a certain percentage of dataset samples.

A.1.3 Cost Assignment

In order to assign costs corresponding to each feature, we conducted a survey to collect the level of overall inconvenience that asking for each feature would cause to subjects. Specifically, we collected survey data from 108 individuals using the Amazon Mechanical Turk framework. Since the data is collected from individuals in the United States, we limited our survey to the same population. Table A.1 presents the questionnaire used in this study.

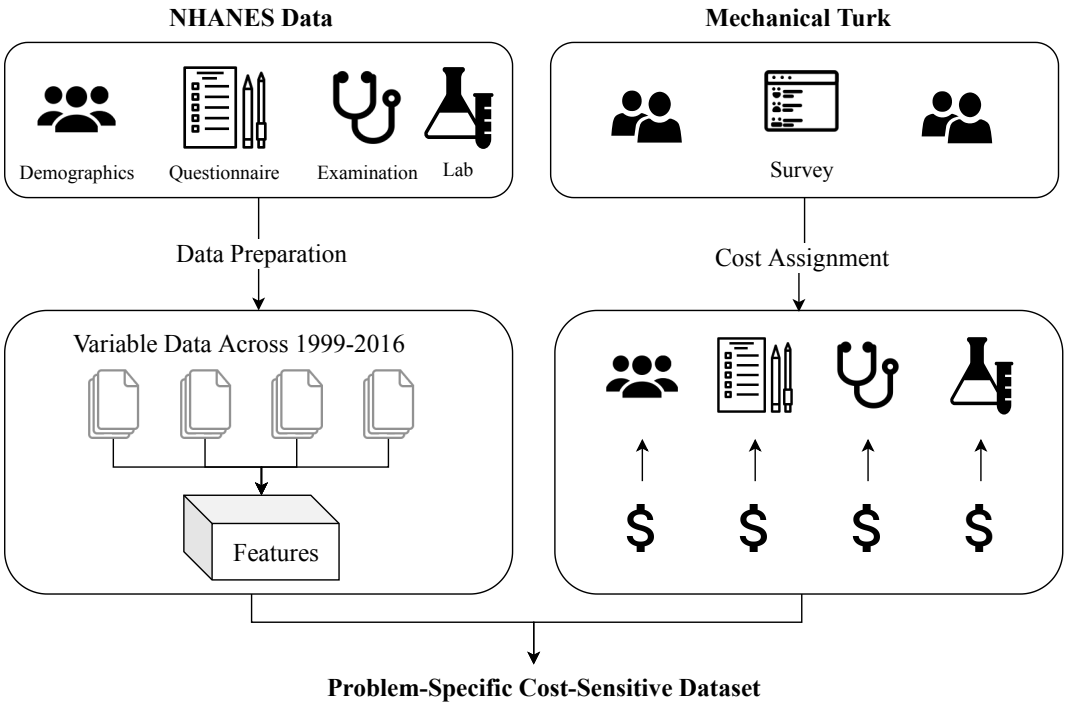


Figure A.1: Visualization of the proposed preprocessing and cost assignment pipeline.

Table A.1: The questionnaire used in this study.

No.	Question	Answer
1	Convenience of answering general demographics related questions (e.g., age, gender, race, etc.)	1...10
2	Convenience of answering general behavioral/life-style related questions (e.g., smoking habits, sleeping habits, alcohol consumption, drug usage etc.)	1...10
3	Convenience of getting typical examinations such as weight, height, or blood pressure measurement	1...10
4	Convenience of taking a blood or urine test at a lab	1...10

Before starting the questionnaire, we asked the turkers to pay attention to the following instructions:

- Please rate each question in terms of the total inconvenience they will cause you (in-

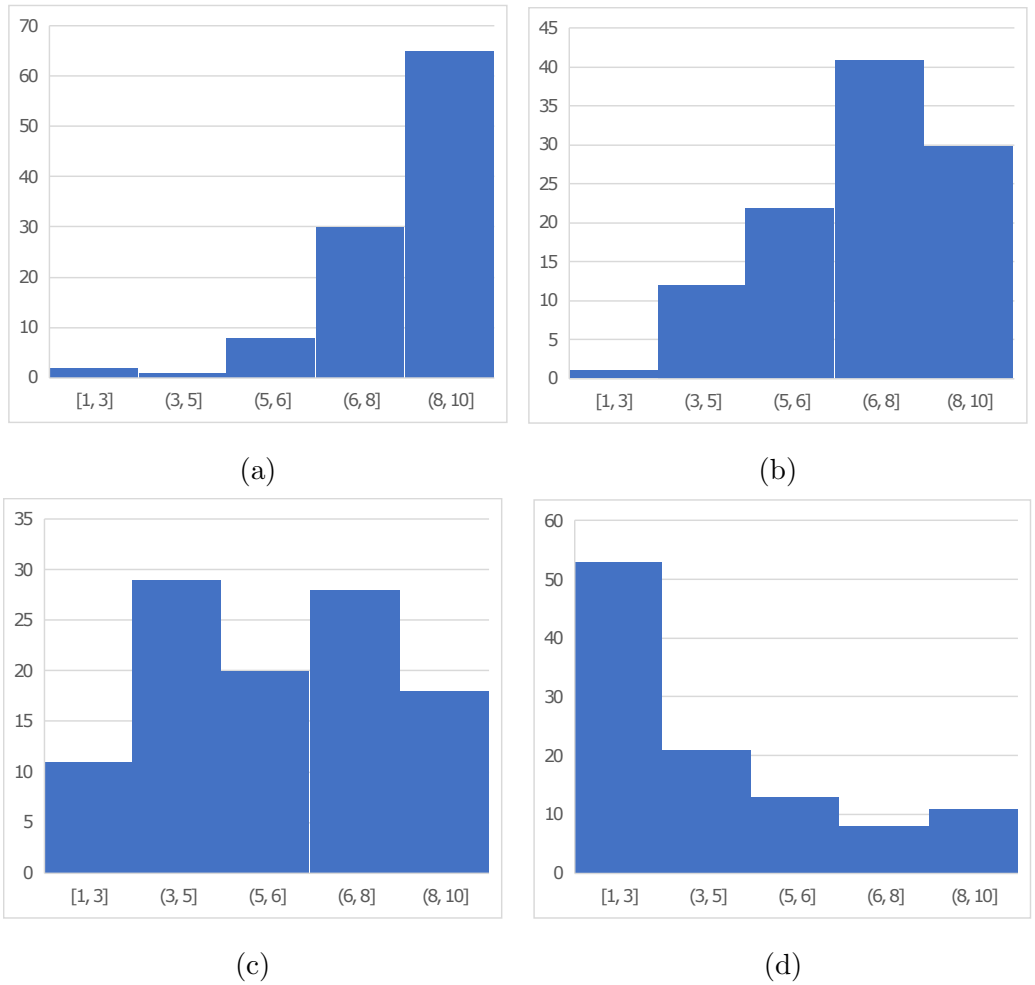


Figure A.2: Distribution of answers for the level of convenience collecting information about: (a) demographics, (b) behavioral/life-style, (c) medical examinations, and (d) lab tests.

cluding time burden, financial cost, discomfort, etc.).

- Assume that each item will provide you with useful health information; however, there is no urgency to do any of these.
- The scale is 1 to 10, 10 being the most convenient. Rate each item based on the relative level of inconvenience.
- After completing the sheet, please review and adjust, if necessary.

The median of survey results for each question is used as a level of convenience for each question. In order to convert these values to cost values (i.e., the higher the more

expensive), we subtract each convenience value from 11 and consider the resulting value as the cost of acquiring features of that category. Accordingly, the final cost for feature categories corresponding to questions 1 to 4 of Table A.1 is determined to be 2, 4, 5 and 9, respectively. See Figure A.1 for an overview of the data preparation methodology and Figure A.2 for the distribution of answers.

A.1.4 Problem Definition

In this chapter, we consider the general scenario of supervised classification using a set of features $\boldsymbol{x}_i \in \mathbb{R}^d$ and ground truth target labels $\tilde{y}_i$. However, initially, the values of features are not available and there is a cost for acquiring each feature $(\boldsymbol{c}_j; 1 \leq j \leq d)$. Consequently, for sample i, at each time step t, we only have access to a partial realization of the feature vector denoted by $\boldsymbol{x}_i^t$ consisting of features that are acquired until t. There might be a maximum budget (B) or a user-defined termination condition (e.g., prediction confidence) which limits the features being acquired eventually.

More formally, we define a mask vector $\boldsymbol{k}_i^t \in \{0,1\}^d$ where each element of $\boldsymbol{k}$ indicates if the corresponding feature is available in $\boldsymbol{x}_i^t$. Using this notation, the total feature acquisition cost at each time step can be represented as

$$C_{total,i}^t = (\boldsymbol{k}_i^t - \boldsymbol{k}_i^0)^T \boldsymbol{c} \quad . \tag{A.1}$$

Intuitively, it measures the total budget spent from the initial state to the current state at time t. Furthermore, we define the feature query operator (q) as

$$\boldsymbol{x}_i^{t+1} = q(\boldsymbol{x}_i^t, j), \text{ where } \boldsymbol{k}_{i,j}^{t+1} - \boldsymbol{k}_{i,j}^t = 1 \quad . \tag{A.2}$$

Note that acquiring a feature at time t results in the transition to a new state, $t+1$, having access to the value of that feature. In this setting, the objective of a cost-sensitive feature acquisition algorithm is to balance the accuracy versus cost trade-off via efficiently acquiring as many features as necessary at test-time.

A.1.5 Sensitivity-Based Approach

Sensitivity-based approaches use trained classifier models and select features that have the most influence on the model predictions. This influence can be used as a utility function which measures the importance of having access to each feature value.

Early *et al.* [EFM16] suggested an exhaustive measurement of expected sensitivity for each feature:

$$\mathbb{E}[U(x^t, j)] = \int p(x_j = r|x^t)U(x^t, x_j = r)dr, \tag{A.3}$$

where $U(x^t, j)$ is the expected utility of acquiring feature j given the feature vector at t, and with the abuse of notation, $U(x^t, x_j = r)$ is the utility of that feature assuming its value after acquisition would be equal to r. It should be noted that as this method requires modeling joint probability distributions as well as integration over feature values, it is not scalable to datasets consisting of many features.

In [KDM18], we suggested an approximation of (A.3) using a binary representation layer in denoising autoencoder architectures:

$$j^t_{sel} = \underset{j\in\{1\dots d\}|k^t_j=0}{argmax} \frac{\sum_{x_j\in RS} |\frac{\partial h(\boldsymbol{x}^t)}{\partial x_j}|\, p(x_j|\boldsymbol{x}^t; h^t)}{c^t_j}, \tag{A.4}$$

where $RS = \{2^{-l}, \dots, 2^{-2}, 2^{-1}, 1\}$ and $h(x^t)$ is the classification function. It is worth noting that this approximation leads to much faster computation based on a single forward and backward evaluation of the network. Specifically, approximating $p(x_j|\boldsymbol{x}^t; h^t)$ terms using a denoising autoencoder with a binary layer and approximating $|\frac{\partial h(\boldsymbol{x}^t)}{\partial x_j}|$ terms via a backpropagation from outputs to the binary input layer.

A.1.6 Reinforcement Learning Approach

The cost-sensitive feature acquisition problem can be formulated as a reinforcement learning problem. In this setting, each state would be the features that are acquired at each point. Additionally, each action would be to pay for a certain feature and to acquire its value, transitioning to a new state. Here, the objective would be to learn a policy function that

results in an efficient feature acquisition. One possible reward function which is frequently used in the literature [JPL17, SHY17] is:

$$r(x_i^t, a) = \begin{cases} -\boldsymbol{c}_j & a \text{ is acquiring feature } j \\ 0 & a \text{ is making a correct prediction} \\ -\lambda & a \text{ is making an incorrect prediction} \end{cases}, \tag{A.5}$$

where λ is a hyperparameter controlling the acquisition cost and prediction accuracy trade-off. In this equation, $-\lambda$ is to penalise incorrect predictions and $-\boldsymbol{c}_j$ is to penalise acquiring each feature according to its cost. Therefore, using higher λ values increases the prediction accuracy while increasing the total cost expenditures.

Alternatively, the variations of model certainty weighted by feature costs can be used to define a reward function [KGK19]:

$$r_{i,j}^t = \frac{\|Cert(\boldsymbol{x}_i^t) - Cert(q(\boldsymbol{x}_i^t, j))\|}{\boldsymbol{c}_j}, \tag{A.6}$$

where $Cert(x)$ represents the prediction certainty[GG16] using a feature vector x. Intuitively, it measures the value of each feature based on the amount which having access to that feature would contribute to making more confident predictions. This method is shown to offer the state of the art results for cost-sensitive feature acquisition at test-time [KGK19]. Furthermore, it is highly scalable and applicable to online stream processing. This can be particularly useful in clinical setups in which prompt decisions are vitally important. For instance, prescribing certain tests and making fast and yet reasonably accurate diagnosis can be life saving.

A.2 Experiments

In order to evaluate and provide baselines for the proposed dataset, in this section, we define specific cost-sensitive classification tasks and present comparison results for several recent cost-sensitive learning methods. Specifically, we compare: *i*) a method based on reinforcement learning where a hyper-parameter is balancing the cost vs. accuracy trade-off

[JPL17], *ii*) Opportunistic Learning (OL) [KGK19] a method based on deep Q-learning with variations of model uncertainty as the reward function, *iii*) a method based on exhaustive measurements of the sensitivity [EFM16], and *iv*) a method based on approximation of sensitivities using denoising autoencoders (FACT) [KDM18]. Table A.2 presents a summary of dataset tasks we defined including the number of instances, features, and classes as well as baseline classification accuracies for each task.

Table A.2: The summary of datasets and experimental settings.

Task	**Instances**	**Features**	**Classes**	**Baseline Accuracy**
Diabetes	92062	45	3	84.2%
Heart Disease	49509	97	2	79.7%
Hypertension	22270	31	2	81.9%

We use PyTorch computational library [PGC17] to train and evaluate multi-layer neural network architectures. Throughout experiments, adaptive momentum (Adam) is used as the optimization algorithm [KB14]. Each experiment took between a few hours to a day on a GPU server. It is worth noting that we normalize dataset features to have zero mean and unit variance prior to our experiments. This normalization permits using the value of zero for missing features during the prediction to act as mean imputation. Regarding the number of layers and hidden neurons, we used a similar number of trainable parameters for OL [KGK19] and RL-Based [JPL17], while due to the inherent differences, we had to use different architectures for FACT [KDM18] and Exhustive [EFM16]. Nonetheless, for each classification task, the compared models reach a similar baseline accuracy (i.e., average accuracy after acquiring all the features). In the following, we provide a brief explanation of each task. For more details about specific features and setups please refer to the Git page[3].

We defined diabetes as the classification objective to predict the blood glucose categories

[3] `https://github.com/mkachuee/Opportunistic`

that fall into the following three categories: normal (blood sugar less than 100 mg/dL), pre-diabetes (blood sugar between 100-125 mg/dL), and diabetes (blood sugar more than 125 mg/dL). We specifically defined and used 45 relevant features from demographics, questionnaire, examinations, and lab tests. Figure A.3 presents a comparison of results based on this task. As it can be seen from this figure, FACT and OL achieve superior results compared to other work. Figure A.4 provides a visualization of feature acquisition order of 50 randomly selected samples for demographic, examination, laboratory, and questionnaire feature categories. In this figure, features that are acquired by OL with more priority are indicated with warmer colors. As it can be inferred from this figure, questionnaire information are usually being acquired with more priority as they provide valuable information at a reasonable cost.

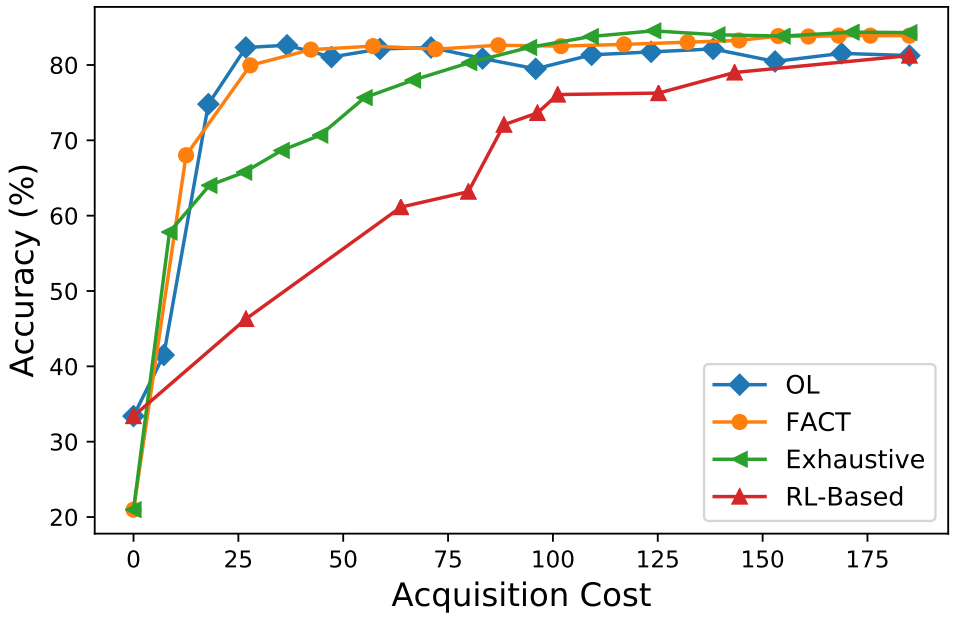


Figure A.3: Accuracy versus cost curve for the diabetes classification task comparing OL [KGK19], FACT [KDM18], Exhaustive [EFM16], and RL-Based [JPL17] methods.

As another task, we consider heart disease classification which is defined as binary classification task. Here, positive samples are individuals that reported any heart disease related issue in their history (e.g., heart attack, heart failure, etc.). For this task, we used the automated feature selection method as explained in the Data Preparation section resulting in 97 features. Figure A.7 shows the relative importance for the 16 most relevant features used in

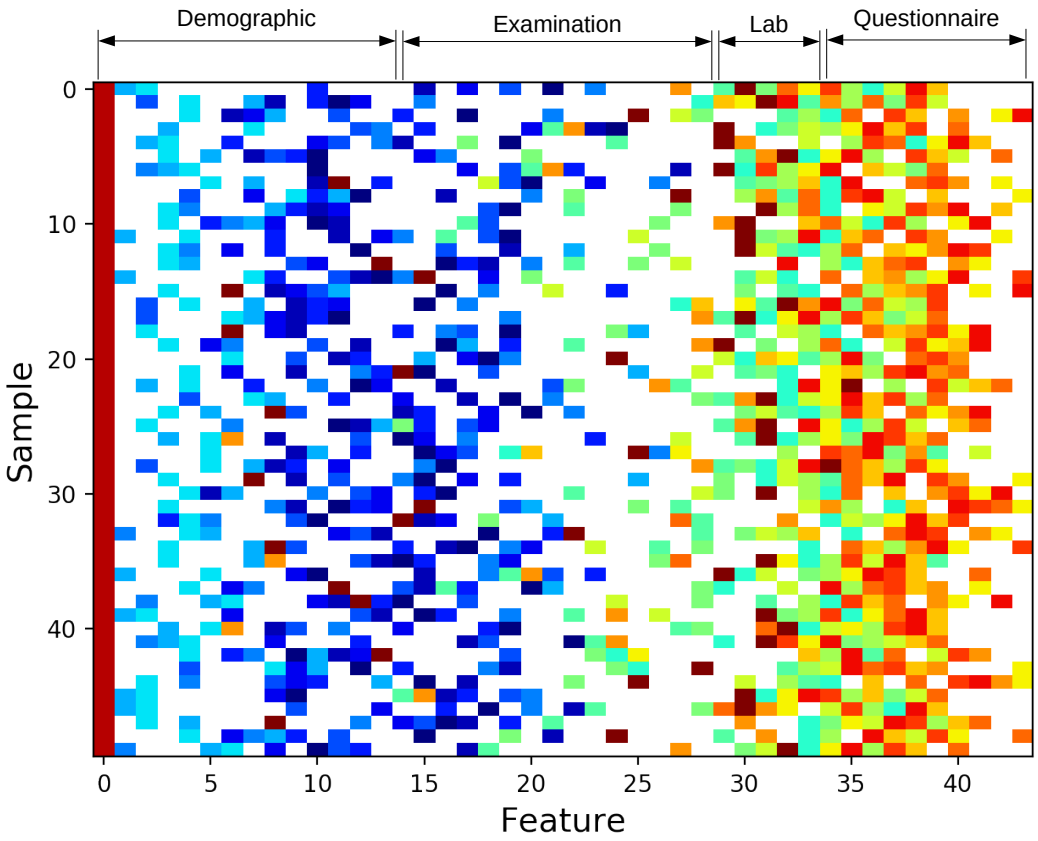


Figure A.4: Visualization of feature acquisition orders using OL [KGK19] method on the diabetes dataset. Warmer colors indicate more priority.

the heart disease classification task. Here, the feature importance is measured based on the relative magnitude of weights for a logistic classifier trained on this dataset. As it can be seen from this figure, the suggested automated variable selection used for this task selects features that are intuitively relevant to heart conditions. Furthermore, Figure A.5 presents the comparison of results for this task. It can be inferred that this task is relatively simple and most approaches were able to achieve a reasonable performance.

At last, we consider the problem of predicting the existence of hypertension condition in individuals. Specifically, we consider subjects with systolic blood pressure of more than 140 mmHg as positive (hypertensive) class. Figure A.6 shows the performance of different methods on this dataset. It can be inferred from this figure that this task is relatively easy and all methods were able to achieve a reasonable performance. The only exception is the

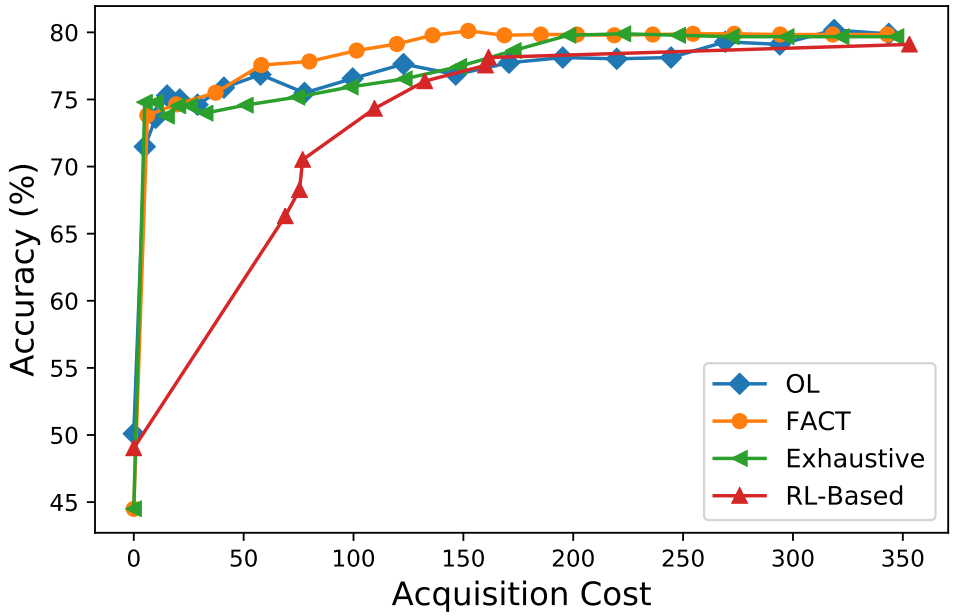


Figure A.5: Accuracy versus cost curve for the heart disease classification task comparing OL [KGK19], FACT [KDM18], Exhaustive [EFM16], and RL-Based [JPL17] methods.

RL-based method which we were not able to find hyper-parameters resulting in cost values in the range of 0 and 5.

A.3 Discussion

Classical approaches to machine learning sought to improve the efficiency and accuracy of prediction but often failed to account for the costs associated with the collection of data and expert labels; the acquisition of some features might incur more costs (monetary and non-monetary) than others. This shortcoming is particularly limiting in the health informatics, where accurate classification often requires an invasive level of information querying. Furthermore, in domains such as medical diagnosis, appropriate data should be collected based on scientific hypothesis, and ground-truth labels may only be provided by highly trained domain experts. Additionally, in many studies, informative features are not scientifically predetermined, and usually, there are many information sources that can be considered as hypothetical relevant features which including all of them is not practical. This is one of the

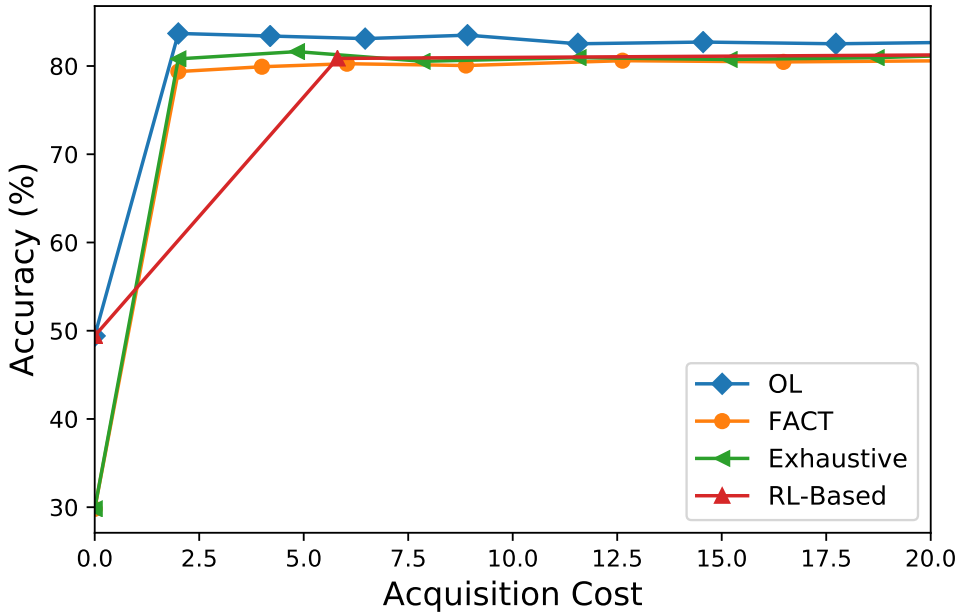


Figure A.6: Accuracy versus cost curve for the hypertension classification task comparing OL [KGK19], FACT [KDM18], Exhaustive [EFM16], and RL-Based [JPL17] methods.

reasons healthcare data remains underutilized. With the explosion of data that the world of IoT brings to the healthcare domain, solutions that are able to efficiently take the unique requirements of smart-health, and scale effectively to the amount of data that is available, are needed. Developing intelligent aggregation methods that leverage available data, enables us to collect the right information at the right time.

In order to address these issues, cost-sensitive and context-aware learning methods were suggested that consider different aspects of a real-world learning-based system. Specifically, addressing all components needed in such a system; including feature acquisition, labeling, model training, and prediction at test-time, each trying to achieve the goal of making accurate predictions efficiently. In this paradigm, information is acquired incrementally based on the value it provides and the cost that should be paid for acquiring it. For instance, in the medical diagnosis use case, while asking for an MRI scan might be more informative than a simple blood test, asking for the blood test might be a better option to start with, based on the information gain per unit of cost for a blood test. This can potentially result in huge

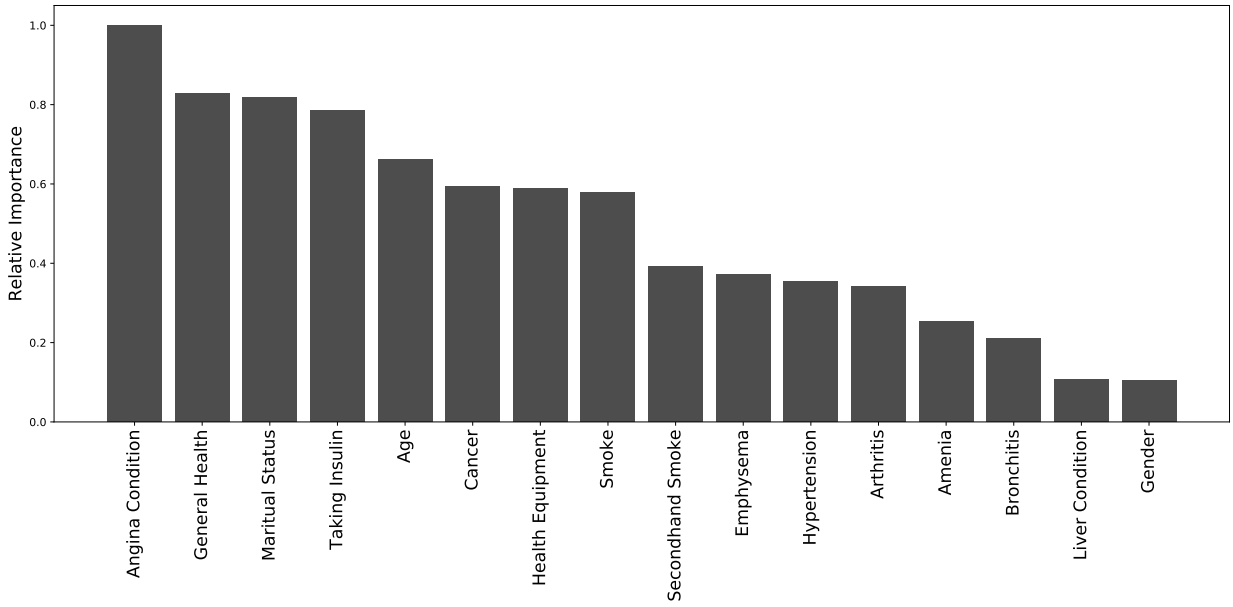


Figure A.7: The relative importance for the 16 most relevant features used in the heart disease classification task.

monetary and non-monetary savings. Finally, in this paradigm, data collection and training are coupled with each other which results in collecting data as much as required while making accurate predictions. This is in contrast to the traditional machine learning setup which collecting a reasonable size of data is assumed to be happening before conducting any analysis.

In this chapter, we explored dynamic and context-aware information acquisition techniques to collect the right piece of information at the right time. The proposed solution, in healthcare settings, enhances the human subject compliance and experience by reducing the cost and inconvenience of medical tests and the data collection. We extensively studied these techniques under the name of "Dynamic Cost-Aware Feature Acquisition"[KDM18] and "Opportunistic Learning"[KGK19]. In this framework, each information piece has a cost which can be predetermined by the user or the expert. In this study, we presented a comparison between the applicability of these methods as well as related work in the literature to health domain problems. Additionally, the scarcity of datasets in healthcare that provide feature costs limited the application of cost-sensitive methods in this domain. We suggested

a methodology for creating such a dataset and published the relevant data and related source code. We hope this study to motivate the development and implementation of cost-sensitive learning techniques for healthcare problems in which the notion of data collection costs, human subject compliance, and patient discomfort are of paramount importance.

REFERENCES

[AAB16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." *arXiv preprint arXiv:1603.04467*, 2016.

[And57] Theodore W Anderson. "Maximum likelihood estimates for a multivariate normal distribution when some observations are missing." *Journal of the american Statistical Association*, **52**(278):200–203, 1957.

[ARZ18] Sanjeev Arora, Andrej Risteski, and Yi Zhang. "Do GANs learn the distribution? some theory and empirics." 2018.

[AWD20] Aliya Aleryani, Wenjia Wang, and Beatriz De La Iglesia. "Multiple Imputation Ensembles (MIE) for Dealing with Missing Data." *SN Computer Science*, **1**:1–20, 2020.

[BET08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. "Speeded-up robust features (SURF)." *Computer vision and image understanding*, **110**(3):346–359, 2008.

[BG10] S van Buuren and Karin Groothuis-Oudshoorn. "mice: Multivariate imputation by chained equations in R." *Journal of statistical software*, pp. 1–68, 2010.

[BM98] Catherine L Blake and Christopher J Merz. "UCI Repository of machine learning databases [http://www. ics. uci. edu/˜ mlearn/MLRepository. html]. Irvine, CA: University of California." *Department of Information and Computer Science*, **55**, 1998.

[CBG00] Diogo Ayres-de Campos, Joao Bernardes, Antonio Garrido, Joaquim Marques-de Sa, and Luis Pereira-Leite. "SisPorto 2.0: a program for automated analysis of cardiotocograms." *Journal of Maternal-Fetal Medicine*, **9**(5):311–318, 2000.

[CC11] Olivier Chapelle and Yi Chang. "Yahoo! learning to rank challenge overview." In *Proceedings of the Learning to Rank Challenge*, pp. 1–24, 2011.

[CCD13] Suming Jeremiah Chen, Arthur Choi, and Adnan Darwiche. "An Exact Algorithm for Computing the Same-Decision Probability." In *IJCAI*, pp. 2525–2531, 2013.

[CCD14] Suming Jeremiah Chen, Arthur Choi, and Adnan Darwiche. "Algorithms and applications for the same-decision probability." *Journal of Artificial Intelligence Research*, **49**:601–633, 2014.

[CCD15] Suming Jeremiah Chen, Arthur Choi, and Adnan Darwiche. "Value of Information Based on Decision Robustness." In *AAAI*, pp. 3503–3510, 2015.

[CDA16a] Gabriella Contardo, Ludovic Denoyer, and Thierry Artières. "Recurrent neural networks for adaptive feature acquisition." In *International Conference on Neural Information Processing*, pp. 591–599. Springer, 2016.

[CDA16b] Gabriella Contardo, Ludovic Denoyer, and Thierry Artieres. "Sequential Cost-Sensitive Feature Acquisition." In *International Symposium on Intelligent Data Analysis*, pp. 284–294. Springer, 2016.

[CLC19] Pengfei Chen, Benben Liao, Guangyong Chen, and Shengyu Zhang. "Understanding and Utilizing Deep Neural Networks Trained with Noisy Labels." *arXiv preprint arXiv:1905.05040*, 2019.

[CXD12] Arthur Choi, Yexiang Xue, and Adnan Darwiche. "Same-decision probability: A confidence measure for threshold-based decisions." *International Journal of Approximate Reasoning*, **53**(9):1415, 2012.

[CXW12] Minmin Chen, Zhixiang Xu, Kilian Weinberger, Olivier Chapelle, and Dor Kedem. "Classifier cascade for minimizing feature evaluation cost." In *Artificial Intelligence and Statistics*, pp. 218–226, 2012.

[CZC10] B Barla Cambazoglu, Hugo Zaragoza, Olivier Chapelle, Jiang Chen, Ciya Liao, Zhaohui Zheng, and Jon Degenhardt. "Early exit optimizations for additive machine learned ranking systems." In *Proceedings of the third ACM international conference on Web search and data mining*, pp. 411–420. ACM, 2010.

[CZZ13] Peng Cao, Dazhe Zhao, and Osmar Zaiane. "An optimized cost-sensitive SVM for imbalanced data learning." In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 280–292. Springer, 2013.

[Dar09] Adnan Darwiche. *Modeling and reasoning with Bayesian networks*. Cambridge university press, 2009.

[DBP16] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. "Adversarially learned inference." *arXiv preprint arXiv:1606.00704*, 2016.

[DG17] Dheeru Dua and Casey Graff. "UCI Machine Learning Repository.", 2017.

[DK17] Dua Dheeru and Efi Karra Taniskidou. "UCI Machine Learning Repository.", 2017.

[EFM16] Kirstin Early, Stephen E Fienberg, and Jennifer Mankoff. "Test time feature ordering with FOCUS: interactive predictions with minimal user burden." In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 992–1003. ACM, 2016.

[EHJ04] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. "Least angle regression." *The Annals of statistics*, **32**(2):407–499, 2004.

[EMF16] Kirstin Early, Jennifer Mankoff, and Stephen E Fienberg. "Dynamic Question Ordering in Online Surveys." *arXiv preprint arXiv:1607.04209*, 2016.

[GAS15] Hassan Ghasemzadeh, Navid Amini, Ramyar Saeedi, and Majid Sarrafzadeh. "Power-aware computing in wearable sensor networks: An optimal feature selection." *IEEE Transactions on Mobile Computing*, **14**(4):800–812, 2015.

[GG16] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning." In *international conference on machine learning*, pp. 1050–1059, 2016.

[GGR02] Russell Greiner, Adam J Grove, and Dan Roth. "Learning cost-sensitive active classifiers." *Artificial Intelligence*, **139**(2):137–174, 2002.

[GPM14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[GPS17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. "On calibration of modern neural networks." *arXiv preprint arXiv:1706.04599*, 2017.

[HDE12] He He, Hal Daumé III, and Jason Eisner. "Cost-sensitive dynamic feature selection." In *ICML Inferning Workshop*, 2012.

[HMK16] He He, Paul Mineiro, and Nikos Karampatziakis. "Active information acquisition." *arXiv preprint arXiv:1602.02181*, 2016.

[HRU17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

[HTS99] Trevor Hastie, Robert Tibshirani, Gavin Sherlock, Michael Eisen, Patrick Brown, and David Botstein. "Imputing missing data for gene expression arrays.", 1999.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[JC07] Shihao Ji and Lawrence Carin. "Cost-sensitive feature acquisition and classification." *Pattern Recognition*, **40**(5):1474–1485, 2007.

[JGP16] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax." *arXiv preprint arXiv:1611.01144*, 2016.

[JK02] Kalervo Järvelin and Jaana Kekäläinen. "Cumulated gain-based evaluation of IR techniques." *ACM Transactions on Information Systems (TOIS)*, **20**(4):422–446, 2002.

[JPL17] Jaromír Janisch, Tomáš Pevnỳ, and Viliam Lisỳ. "Classification with Costly Features using Deep Reinforcement Learning." *arXiv preprint arXiv:1711.07364*, 2017.

[KB14] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*, 2014.

[KBF12] Sergey Karayev, Tobias Baumgartner, Mario Fritz, and Trevor Darrell. "Timely object recognition." In *Advances in Neural Information Processing Systems*, pp. 890–898, 2012.

[KCL19] Pasha Khosravi, YooJung Choi, Yitao Liang, Antonio Vergari, and Guy Van den Broeck. "On tractable computation of expected predictions." In *Advances in Neural Information Processing Systems*, pp. 11169–11180, 2019.

[KDM18] Mohammad Kachuee, Sajad Darabi, Babak Moatamed, and Majid Sarrafzadeh. "Dynamic Feature Acquisition Using Denoising Autoencoders." *IEEE transactions on neural networks and learning systems*, 2018.

[KGK19] Mohammad Kachuee, Orpaz Goldstein, Kimmo Kärkkäinen, Sajad Darabi, and Majid Sarrafzadeh. "Opportunistic Learning: Budgeted Cost-Sensitive Learning from Data Streams." 2019.

[KH09] Alex Krizhevsky and Geoffrey Hinton. "Learning multiple layers of features from tiny images." Technical report, Citeseer, 2009.

[KHM17] Mohammad Kachuee, Anahita Hosseini, Babak Moatamed, Sajad Darabi, and Majid Sarrafzadeh. "Context-aware feature query to improve the prediction performance." In *Signal and Information Processing (GlobalSIP), 2017 IEEE Global Conference on*, pp. 838–842. IEEE, 2017.

[KKG19] Mohammad Kachuee, Kimmo Karkkainen, Orpaz Goldstein, Davina Zamanzadeh, and Majid Sarrafzadeh. "Nutrition and Health Data for Cost-Sensitive Learning." *arXiv preprint arXiv:1902.07102*, 2019.

[KKM17] Mohammad Kachuee, Mohammad Mahdi Kiani, Hoda Mohammadzade, and Mahdi Shabany. "Cuffless Blood Pressure Estimation Algorithms for Continuous Health-Care Monitoring." *IEEE Transactions on Biomedical Engineering*, **64**(4):859–869, 2017.

[KMH17] Mohammad Kachuee, Lisa D Moore, Tali Homsey, Hamidreza Ghasemi Damavandi, Babak Moatamed, Anahita Hosseini, Ruyi Huang, James Leiter, Daniel Lu, and Majid Sarrafzadeh. "An Active Learning Based Prediction of Epidural Stimulation Outcome in Spinal Cord Injury Patients Using Dynamic Sample Weighting." In *Healthcare Informatics (ICHI), 2017 IEEE International Conference on*, pp. 478–483. IEEE, 2017.

[KVC20] Pasha Khosravi, Antonio Vergari, YooJung Choi, Yitao Liang, and Guy Van den Broeck. "Handling missing data in decision trees: A probabilistic approach." *arXiv preprint arXiv:2006.16341*, 2020.

[KW13] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114*, 2013.

[KYR11] Balaji Krishnapuram, Shipeng Yu, and R Bharat Rao. *Cost-sensitive Machine Learning*. CRC Press, 2011.

[LBC17] Shuang Liu, Olivier Bousquet, and Kamalika Chaudhuri. "Approximation and convergence properties of generative adversarial learning." In *Advances in Neural Information Processing Systems*, pp. 5545–5553, 2017.

[LCB98] Yann LeCun, Corinna Cortes, and Christopher JC Burges. "The MNIST database of handwritten digits.", 1998.

[Lew97] David D Lewis. "Reuters-21578 text categorization test collection, distribution 1.0." 1997.

[LJM19] Steven Cheng-Xian Li, Bo Jiang, and Benjamin Marlin. "Learning from Incomplete Data with Generative Adversarial Networks." In *International Conference on Learning Representations*, 2019.

[LR19] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. Wiley, 2019.

[LXL17] Meng Liu, Chang Xu, Yong Luo, Chao Xu, Yonggang Wen, and Dacheng Tao. "Cost-Sensitive Feature Selection via F-Measure Optimization Reduction." In *AAAI*, pp. 2252–2258, 2017.

[MF18] Pierre-Alexandre Mattei and Jes Frellsen. "missIWAE: Deep Generative Modelling and Imputation of Incomplete Data." *arXiv preprint arXiv:1812.02633*, 2018.

[MHZ14] Fan Min, Qinghua Hu, and William Zhu. "Feature selection with test cost constraint." *International Journal of Approximate Reasoning*, **55**(1):167–179, 2014.

[MKK18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. "Spectral normalization for generative adversarial networks." *arXiv preprint arXiv:1802.05957*, 2018.

[MKS13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602*, 2013.

[MKS15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. "Human-level control through deep reinforcement learning." *Nature*, **518**(7540):529, 2015.

[MM01] George B Moody and Roger G Mark. "The impact of the MIT-BIH arrhythmia database." *IEEE Engineering in Medicine and Biology Magazine*, **20**(3):45–50, 2001.

[Mur18] Jared S Murray et al. "Multiple imputation: a review of practical and theoretical findings." *Statistical Science*, **33**(2):142–159, 2018.

[NDR13] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. "Learning with noisy labels." In *Advances in neural information processing systems*, pp. 1196–1204, 2013.

[Nea04] Richard E Neapolitan et al. *Learning bayesian networks*, volume 38. Pearson Prentice Hall Upper Saddle River, NJ, 2004.

[NH10] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines." In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[nha18] "National Health and Nutrition Examination Survey.", 2018.

[NJ09] Thomas Dyhre Nielsen and Finn Verner Jensen. *Bayesian networks and decision graphs*. Springer Science & Business Media, 2009.

[NOG18] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. "Handling incomplete heterogeneous data using VAEs." *arXiv preprint arXiv:1807.03653*, 2018.

[NS17] Feng Nan and Venkatesh Saligrama. "Adaptive Classification for Prediction Under a Budget." In *Advances in Neural Information Processing Systems*, pp. 4727–4737, 2017.

[PGC17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. "Automatic differentiation in PyTorch." In *NIPS-W*, 2017.

[PKU15] Erman Pattuk, Murat Kantarcioglu, Huseyin Ulusoy, and Bradley Malin. "Privacy-aware dynamic feature selection." In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pp. 78–88. IEEE, 2015.

[RKK20] Seunghyoung Ryu, Minsoo Kim, and Hongseok Kim. "Denoising Autoencoder-Based Missing Value Imputation for Smart Meters." *IEEE Access*, **8**:40656–40666, 2020.

[RLA14] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. "Training deep neural networks on noisy labels with bootstrapping." *arXiv preprint arXiv:1412.6596*, 2014.

[ROS16] Jorge-L Reyes-Ortiz, Luca Oneto, Albert Sama, Xavier Parra, and Davide Anguita. "Transition-aware human activity recognition using smartphones." *Neurocomputing*, **171**:754–767, 2016.

[Rub76] Donald B Rubin. "Inference and missing data." *Biometrika*, **63**(3):581–592, 1976.

[Sch81] Jeff Schlimmer. "Mushroom records drawn from The Audubon Society field guide to north American mushrooms." *GH Lincoff (Pres), New York*, 1981.

[SG02] Joseph L Schafer and John W Graham. "Missing data: our view of the state of the art." *Psychological methods*, **7**(2):147, 2002.

[SHY17] Hajin Shim, Sung Ju Hwang, and Eunho Yang. "Why Pay More When You Can Pay Less: A Joint Learning Framework for Active Feature Acquisition and Classification." *arXiv preprint arXiv:1709.05964*, 2017.

[SKS20] Marek Śmieja, Maciej Kołomycki, Łukasz Struski, Mateusz Juda, and Mário A. T. Figueiredo. "Can auto-encoders help with filling missing data?" In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[SLY15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. "Learning structured output representation using deep conditional generative models." In *Advances in neural information processing systems*, pp. 3483–3491, 2015.

[SS95] Peter K. Sharpe and RJ Solly. "Dealing with missing values in neural network-based diagnostic systems." *Neural Computing & Applications*, **3**(2):73–77, 1995.

[SZS13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks." *arXiv preprint arXiv:1312.6199*, 2013.

[TS13] Kirill Trapeznikov and Venkatesh Saligrama. "Supervised sequential classification under budget constraints." In *Artificial Intelligence and Statistics*, pp. 581–589, 2013.

[TZA17] Cao Truong Tran, Mengjie Zhang, Peter Andreae, and Bing Xue. "Multiple imputation and genetic programming for classification with incomplete data." In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 521–528, 2017.

[VJ04] Paul Viola and Michael J Jones. "Robust real-time face detection." *International journal of computer vision*, **57**(2):137–154, 2004.

[VLB08] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. "Extracting and composing robust features with denoising autoencoders." In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM, 2008.

[Wil97] Christopher KI Williams. "Computing with infinite networks." In *Advances in neural information processing systems*, pp. 295–301, 1997.

[WLZ18] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. "High-resolution image synthesis and semantic manipulation with conditional gans." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8798–8807, 2018.

[XKW14] Zhixiang Eddie Xu, Matt J Kusner, Kilian Q Weinberger, Minmin Chen, and Olivier Chapelle. "Classifier cascades and trees for minimizing feature evaluation cost." *Journal of Machine Learning Research*, **15**(1):2113–2144, 2014.

[XWC12] Zhixiang Xu, Kilian Weinberger, and Olivier Chapelle. "The greedy miser: Learning under test-time budgets." *arXiv preprint arXiv:1206.6451*, 2012.

[YJV18] Jinsung Yoon, James Jordon, and Mihaela Van Der Schaar. "Gain: Missing data imputation using generative adversarial nets." *arXiv preprint arXiv:1806.02920*, 2018.

[ZGM18] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. "Self-attention generative adversarial networks." *arXiv preprint arXiv:1805.08318*, 2018.

[ZQL05] Shichao Zhang, Zhenxing Qin, Charles X Ling, and Shengli Sheng. "" Missing is useful": missing values in cost-sensitive decision trees." *IEEE transactions on knowledge and data engineering*, **17**(12):1689–1693, 2005.