Predicting Performance Run-time Metrics in Fog Manufacturing using Multi-task Learning

Vignesh Raja Nallendran

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
in
Industrial and Systems Engineering

Ran Jin, Chair
Subhash C. Sarin
Xinwei Deng

February 3, 2021

Blacksburg, VA

Keywords: Fog computing, Fog manufacturing, Multi-task learning, Run-time metrics

Predicting Performance Run-time Metrics in Fog Manufacturing using Multi-task Learning

Vignesh Raja Nallendran

ABSTRACT

The integration of Fog-Cloud computing in manufacturing has given rise to a new paradigm called Fog manufacturing. Fog manufacturing is a form of distributed computing platform that integrates Fog-Cloud collaborative computing strategy to facilitate responsive, scalable, and reliable data analysis in manufacturing networks. The computation services provided by Fog-Cloud computing can effectively support quality prediction, process monitoring, and diagnosis efforts in a timely manner for manufacturing processes. However, the communication and computation resources for Fog-Cloud computing are limited in Fog manufacturing. Therefore, it is significant to effectively utilize the computation services based on the optimal computation task offloading, scheduling, and hardware autoscaling strategies to finish the computation tasks on time without compromising on the quality of the computation service. A prerequisite for adapting such optimal strategies is to accurately predict the run-time metrics (e.g., Time-latency) of the Fog nodes by capturing their inherent stochastic nature in real-time. It is because these run-time metrics are directly related to the performance of the computation service in Fog manufacturing. Specifically, since the computation flow and the data querying activities vary between the Fog nodes in practice. The run-time metrics that reflect the performance in the Fog nodes are heterogenous in nature and the performance cannot be effectively modeled through traditional predictive analysis. In this thesis, a multi-task learning methodology is adopted to predict the run-time metrics that reflect performance in Fog manufacturing by addressing the heterogeneities among the Fog nodes. A Fog manufacturing testbed is employed to evaluate the prediction accuracies of the proposed model

and benchmark models. The proposed model can be further extended in computation tasks offloading and architecture optimization in Fog manufacturing to minimize the time-latency and improve the robustness of the system.

Predicting Performance Run-time Metrics in Fog Manufacturing using Multi-task Learning

Vignesh Raja Nallendran

GENERAL AUDIENCE ABSTRACT

Smart manufacturing aims at utilizing Internet of things (IoT), data analytics, cloud computing, etc. to handle varying market demand without compromising the productivity or quality in a manufacturing plant. To support these efforts, Fog manufacturing has been identified as a suitable computing architecture to handle the surge of data generated from the IoT devices. In Fog manufacturing computational tasks are completed locally through the means of interconnected computing devices called Fog nodes. However, the communication and computation resources in Fog manufacturing are limited. Therefore, its effective utilization requires optimal strategies to schedule the computational tasks and assign the computational tasks to the Fog nodes. A prerequisite for adapting such strategies is to accurately predict the performance of the Fog nodes. In this thesis, a multi-task learning methodology is adopted to predict the performance in Fog manufacturing. Specifically, since the computation flow and the data querying activities vary between the Fog nodes in practice. The metrics that reflect the performance in the Fog nodes are heterogenous in nature and cannot be effectively modeled through conventional predictive analysis. A Fog manufacturing testbed is employed to evaluate the prediction accuracies of the proposed model and benchmark models. The results show that the multi-task learning model has better prediction accuracy than the benchmarks and that it can model the heterogeneities among the Fog nodes. The proposed model can further be incorporated in scheduling and assignment strategies to effectively utilize Fog manufacturing's computational services.

ACKNOWLEDGMENT

I would like to start by thanking my advisor Dr Ran Jin for providing me an opportunity to work under him. His valuable inputs have been beneficial both academically and professionally, and his continued inputs and value addition for my research is something that I am grateful for. I would like to thank Dr Subhash Sarin and Dr Xinwei Deng for being part of my committee and guiding me through my master's journey.

I would especially like to thank Mr. Lening Wang, who initially helped me in understanding Fog manufacturing and continued giving valuable suggestions and course corrections during my research. I am grateful for Ms. Yutong Zhang for helping me understand the data for my research and Mr. Xiaoyu Chen for giving me valuable suggestions on my model. I would like to thank all my fellow research members for welcoming me into the Data science and Visualization lab group.

DEDICATION

My journey here at Virginia Tech is all thanks to my parents who have constantly supported me be it financially or otherwise. And most importantly I would like to thank God for his continued grace.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Motivation

Fog manufacturing is an emerging manufacturing platform that integrates the Fog-Cloud collaborative computation services and Cyber-physical systems (CPS) to better support various data analytics in practice [1-4]. To effectively organize and implement the different types of data analytics in manufacturing, computational pipelines have been employed to flexibly demonstrate the on demand computational requirements [5]. A computational pipeline is a sequence of pipeline components and method options for computational tasks that are broken down into sub-steps [5]. There are different computation tasks in manufacturing, such as time series modelling, predictive analytics, etc. Therefore, the pipelines will have different components and method options. My thesis is focused on data driven modelling pipelines, in particular simple data analytics pipelines. A simple data analytics pipeline is a sequential combination of sub-steps (e.g., feature extraction, parameter tuning, modeling, etc.) as shown in Figure 1, where selecting different sub-steps leads to the desired computational flow [6]. It is worth mentioning that the pipelines in manufacturing are not limited to the examples shown in Figure 1. Ideally the sub-step options are large enough to reflect the various computational requirements in a manufacturing facility.

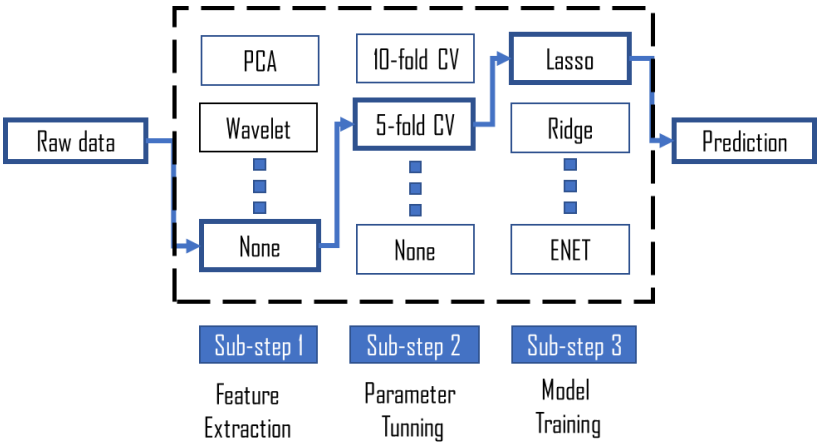

Figure 1: Example of a simple data analytics pipeline

By efficiently analyzing the real-time data collected from manufacturing processes via the pipelines, Fog manufacturing can effectively support time-critical decisions based on the data analysis result with a satisfied time-latency and communication bandwidth utilization [4]. These advantages are accomplished by the Fog-cloud computation service in Fog manufacturing, which can decompose and distribute the computation tasks (i.e., sub-steps) to interconnected and geographically distributed devices (i.e., Fog nodes) that are capable of processing and analyzing the data locally [7].

Compared with a Cloud only computing service which needs to firstly transfer all data to the Cloud then perform the computation, the Fog-Cloud computing service can significantly improve the responsiveness and reliability of the data analysis service by properly offloading (i.e., assigning) the computation tasks to the Fog nodes [13-15]. In addition, compared to Cloud computing, Fog manufacturing can also be scaled up cost-effectively based on the dynamic computation requirements [1,12].

To clearly demonstrate the structure of Fog manufacturing, a schematic diagram is shown in Figure 1. In Figure 1, there are three layers in this Fog manufacturing platform. The first layer is the facility layer, which includes manufacturing systems, robotic arms, and other Internet of Things (IoT) devices with corresponding sensor systems to collect real-time data from the production lines. To implement the computation service for the collected data, the data are efficiently transmitted to the second layer (i.e., Fog layer) via routers and gateways [4,7,8]. The Fog layer consists of geographically distributed and interconnected computation and networking resources. These Fog nodes are capable of conducting computation tasks that are not computationally intensive (e.g., data structuring, pre-processing, data cleaning etc. [7-9]), and will upload the intermediate results to the Cloud for further data analysis. Moreover, the Fog node can also collect the results from the Cloud and effectively transfer the real-time decision making to specific manufacturing system [11,12]. Finally, by collecting the processed data from the Fog nodes, the Cloud server can effectively perform the complex data analytics efforts to comprehensively optimize the manufacturing processes based on the customers' requirements [13].

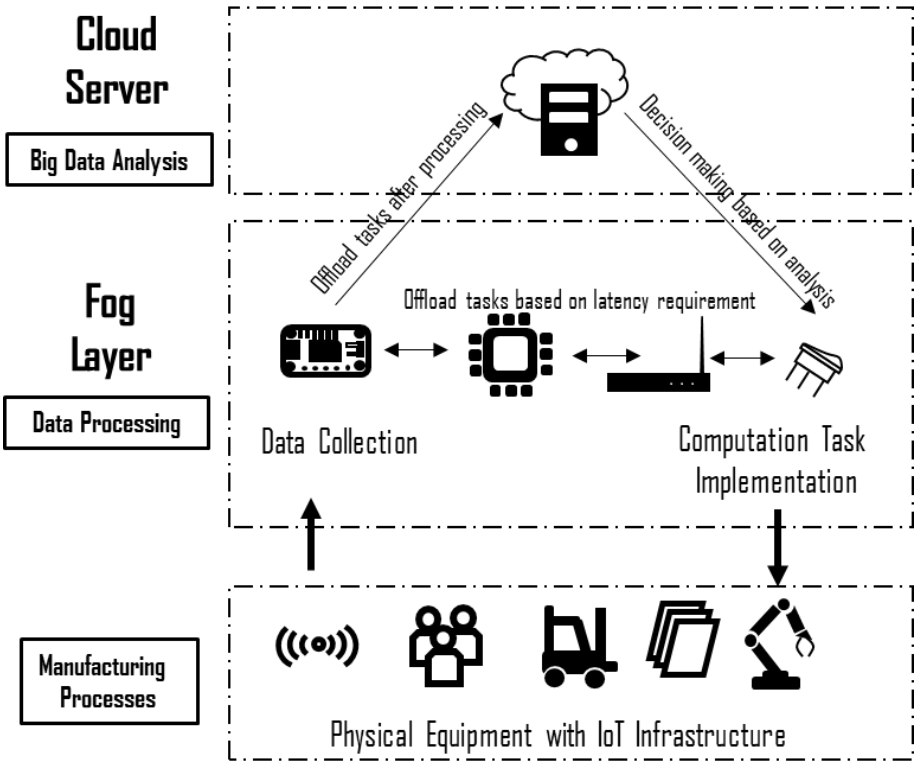

Figure 2: A Schematic diagram for Fog manufacturing

Even though there are many advantages in utilizing Fog manufacturing, there are still some challenges that restrict its wide deployment. One major advantage of Fog manufacturing is implementing the computation tasks in a distributed manner, rather than transmitting data to a centralized server for analysis [12,13]. However, computation and communication resources in Fog manufacturing are limited. To effectively deploy the computation tasks in Fog manufacturing, it is vital to properly schedule computation tasks, assign tasks to the proper Fog nodes at the right time (i.e., offloading), and automatically add or remove resources based on the computation requirements (i.e., autoscaling) [17]. For computation task scheduling and offloading in Fog manufacturing, predictive offloading strategies have garnered recent research interest [18,19,21]. However, optimized offloading decisions rely on the information from run-time metrics of each computation task, such as the time-latency (i.e., total time taken for a computation task on the

specific Fog node) [21,22]. Hence, there is a need to accurately predict the run-time metrics of each computation task before offloading the task to specific Fog nodes. Similarly, in computation scale-up, it is also necessary to know the run-time metrics to proactively determine when to add or remove computation resources in Fog manufacturing [16]. Therefore, it is significant to predict run-time metrics that reflect the performance of Fog nodes before the implementation of each computational task.

1.2 Research Objective and Challenges

The objective of this thesis is to propose a model that can effectively predict the selected run-time metrics on each device in Fog manufacturing. Specifically, Time-latency, CPU utilization, and Bandwidth utilization are studied as the run-time metrics in this work [20]. Time-latency is defined as the total time taken to finish the computation task on a specific Fog node. It is also the key performance indicator to evaluate the responsiveness in Fog manufacturing. CPU utilization is defined as the computation capacity used divided by the total available computation capacity. Depending on the objective for offloading, CPU utilization is either preferred to be high or low. Download utilization indicates the rate of data transfer for downloading, lower the download bandwidth utilization more is the capacity for transferring more data. Apart from the three run-time metrics stated above, Double redundancy is another run-time metric that reflects the reliability of the Fog nodes. Since it is calculated as twice the Time-latency in a Fog node, double redundancy is not separately predicted.

In the literature, to effectively predict the run-time metrics defined above, information generated from the data analytics pipelines and historical *in situ* computation conditions on each Fog node have been identified as important predictors [20]. The hardware configurations of the Fog nodes

are expected to significantly affect the run-time metrics performance. Generally speaking, the Fog nodes have different hardware configurations (e.g., GPU units, CPU units, FPGA units, etc.) in a manufacturing facility. These differences in the hardware can be quantified by introducing new predictor variables that describe the configuration of the Fog node on which a computational task was offloaded.

In this thesis, the Fog nodes have the same hardware configuration (i.e., Raspberry pie 3 units). Instead, the Fog nodes might not have the same computational flow or data query strategy when executing the computation tasks, the relationship between the run-time metrics and the predictors might vary. As shown in Figure 3, the two Fog nodes have identical hardware configuration and performance, but over time the difference in the computational workflow represented by the blue and orange blocks leads to varying accumulative effects in their hardware (e.g., CPU cache, Memory bandwidth). This causes the relationship between the run-time metrics and identified predictor variables to vary between the two Fog nodes, resulting in heterogenous run-time performances. Moreover, modeling these heterogenous Fog nodes in one model would lead to poor prediction accuracy and would negatively affect offloading and computation scaling decisions.

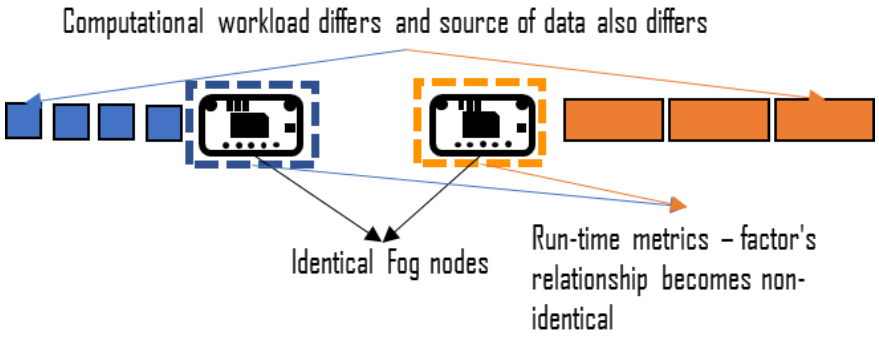

Figure 3: Heterogenous run-time performances of Fog nodes with identical hardware configurations due to different computation workloads and data sources

1.3 Proposed Methodology

In this thesis, to tackle the challenges discussed above, the regularized multi-task learning framework [28,29] is adopted to predict run-time metrics effectively and accurately in Fog manufacturing. Specifically, multi-task learning is employed in this study to efficiently model the data heterogeneity in the run-time metrics present among the Fog nodes. Since the computation flow that the Fog nodes execute are similar-but-non-identical, their models are expected to share similar model coefficient structure (i.e., variable selection results) between each other. This is realized by penalizing on the $l_{2,1}$ and l_2 norms for achieving similarity and sparsity in the model coefficients [29]. The methodology is validated in a Fog manufacturing testbed. Based on the results, the multi-task learning model is shown to have better prediction accuracy by comparing with existing benchmark methods such as linear regression [39], Lasso regression [39] and Random forest regression [40]. Furthermore, the variable selection results are visualized to interpret the heterogeneous characteristics among the Fog nodes.

CHAPTER 2: LITERATURE REVIEW

In this Chapter, the background of Fog manufacturing, existing offloading and scalability techniques in Fog computing, models to predict run-time metrics in Fog computing, and models to address the heterogeneity are reviewed in detail. Beginning with the Fog manufacturing, current research papers have identified Fog computing as a suitable platform for IoT applications and services such as smart grid, smart cities, connected vehicles, and virtual sensors and network actuators, where Cloud computing alone cannot meet the latency requirements for delay sensitive computational tasks [7].

Fog computing is viewed as a platform solution to facilitate real-time data processing with a reduced requirement in bandwidth [8,9]. Consequently, the Fog computing paradigm that extends the Cloud to the Edge has garnered attention in manufacturing. Implementation of Fog computing in manufacturing (i.e., Fog manufacturing) seeks to alleviate traffic to the Cloud and facilitate real-time decision making by processing data locally in a distributed manner. This helps in proactively predicting machine downtime, responding to varying demand, and effectively identifying the root cause of a manufacturing defect [1,2]. In an extension of Fog computing's application in manufacturing, Qi et al. discussed how Fog computing technology can be used to realize a system-level cyber-physical system (CPS) and Digital twin (DT) in smart manufacturing [10]. With these beneficial characteristics, Fog manufacturing has the potential in filling the gap in Cloud computing to respond to industrial requirements. However, implementing Fog computing in smart manufacturing has its challenges, these include 1) heterogeneous nature of data processed, 2) architecture design to allocate resources, and 3) scalability. A starting point to address these

challenges would be to model the computation and communication performance variables for analysis in Fog manufacturing [4].

On the other hand, offloading methods in Fog manufacturing are essential in ensuring the effective use of Fog computing. In Fog manufacturing, a system of sensors and actuators collects data through smart gateways and send them to Fog nodes that support distributed data analytics for real-time decision making [2,11,12]. Since the network resources are limited for distributing tasks to the Fog nodes, scheduling tasks, and knowing when to assign tasks (offloading) becomes critical in ensuring reduced delay and optimizing the decision-making process [13-15]. Another important advantage in Fog manufacturing is its scalability. Based on the computation requirements, the Fog manufacturing can be scaled up or down rapidly and cost-effectively [1]. There are several algorithms proposed to automatically scale (i.e., autoscaling) the computation service based on the requirements of the computation task [16,17]. However, a prerequisite for implementing effective scheduling, offloading, and autoscaling strategies are to predict the run-time metrics of Fog nodes [16,18,19]. In the current literature, to facilitate the prediction of run-time metrics in Fog computing, performance run-time metrics, and corresponding process variables have been established. For example, a systems informatics approach was adapted by Zhang et al. to determine the process variables affecting performance run-time metrics, such as Time-latency and double redundancy [20]. These performance metrics were used as benchmarks to compare performance in Cloud and Fog-Cloud computing. Without accurately predicting the performance run-time metrics, we cannot utilize the fog manufacturing architecture effectively.

In the current literature, there are many predictive models proposed to predict the run-time metrics in the manufacturing system, which are needed to improve the Fog computing performance. Luong et al. proposed a predictive autoscaling method by predicting CPU metrics and throughput metrics

in real-time to predict bust load and short-term future value [16]. Gao et al. predicted workload in Fog nodes to propose a dynamic offloading technique that minimized energy consumption [19]. In a paper by Chen et al., an offloading strategy based on predicting bandwidth and computation using the ARIMA model resulted in better task completion [21]. Patman et al. used machine learning algorithms to predict time-latency and concluded that their predictive models show higher accuracy than discrete calculation approaches [22]. In Fog computing, the Fog nodes handle different computational tasks and databases for storage that leads to process heterogeneity [23]. However, existing predictive models have not addressed the heterogenous characteristics in the run-time metrics among the Fog nodes. Therefore, it may lead to inaccurate prediction results, resulting in ineffective utilization of the Fog computing service.

To model heterogeneity among the Fog nodes, it is necessary to find ways of sharing knowledge between the Fog nodes and improve the overall prediction accuracy. In current literature, models that leverage related tasks to induce the required knowledge transfer exist. For example, Baxter introduced a theoretical Bayesian model that samples from related tasks to induce bias. The bias, in this case, relates to the desired domain knowledge that is reflected in the model coefficients [24]. Caruana introduced the multi-task learning algorithm based on kernel regression and k-nearest neighbors that used related tasks in parallel to achieve the desired bias [25]. Apart from capturing heterogeneity in tasks, modeling multiple related tasks simultaneously produces a better result in terms of variable selection and prediction accuracy than modeling tasks individually. For example, Huang et al. utilized a multi-task sparse Bayesian model that modeled related tasks along with the primary task to get better prediction in structural stiffness [26]. Multi-task Lasso regression was used by Lina et al. the resulting prediction showcased accuracy better than existing benchmark methods [27]. Generalization in the prediction model is achieved by forcing tasks to

share similar coefficient values. In the multi-task Lasso, the values of the input variables remain the same and the tasks are defined as varying output variables. However, in the case of modeling for performance in Fog computing, the value of both the input variables (*in situ* variables) and output variables (run-time metrics) varies across individual Fog nodes, where the task is to predict performance in individual Fog nodes.

A form of regularized multi-task learning is more feasible in this given case [28, 29]. In regularized multi-task learning, the tasks to be modeled share the same process and performance variables, but the context and values of these parameters differ. But these existing methods have not been utilized in the context of predicting performance in Fog manufacturing. In this paper, a form of regularized multi-task learning method is utilized to predict the run-time metrics which are related to the performance and reliability of each Fog nodes in Fog manufacturing.

CHAPTER 3: METHODOLOGY

In this chapter, the details of proposed regularized multi-task learning model is discussed. Specifically, the notations employed in this study are summarized in Table 1. Without the loss of generality, we consider modelling the run-time metrics for an individual Fog node as a single task. This is because the relationship between the performance and predictor variables used for modeling the performance of Fog nodes are non-identical. In this study, a sample for each task is collected when a sub-step from a pipeline is offloaded to a specific Fog node, and run-time metrics (i.e., Time-latency, CPU utilization, Download utilization) for this Fog node are treated as the performance variables in modeling.

Table 1: Notations in the Proposed Multi-task Learning Model

Notation	Description
n, p, t	Number of samples, predictors, and tasks
Y_i	$\mathbb{R}^{\{n_i \times \mathbf{1}\}}$, Response variable for task i
X_i	$\mathbb{R}^{\{n_i \times \boldsymbol{p}\}}$, Predictor variables for task i
W_i	Coefficient of model for task i
λ_1, λ_2	Tuning parameter for Similarity and sparsity
C_i	Normally distributed constant

$L(\cdot)$	Mean squared error loss function
$Y = [\boldsymbol{y}_1^{\mathrm{T}}, \ldots, \boldsymbol{y}_t^{\mathrm{T}}]$	Response variable for all tasks
$X = [\boldsymbol{x}_1^{\mathrm{T}}, \ldots, \boldsymbol{x}_t^{\mathrm{T}}]$	Predictor variable for all tasks
$\boldsymbol{\beta}_i$	$\mathbb{R}^{\{p \times 1\}}$,Model coefficient for task i
$\boldsymbol{B} = [\boldsymbol{\beta}_1^{\mathrm{T}}, \ldots, \boldsymbol{\beta}_t^{\mathrm{T}}]$	Combined model coefficient of all t tasks

3.1 Assumptions

The assumptions for the proposed multi-task learning include: (1) A linear model structure is adequate to model the relationship between the run-time metrics and predictors. (2) The model coefficients among different Fog nodes are similar-but-non-identical. It is because the Fog nodes have identical hardware build and similar computation tasks offloaded to the Fog nodes. This similarity can be captured in the variable selection effort where the Fog nodes share similar model coefficient structure. These two assumptions will be validated in the case study.

3.2 The multi-task learning model

Without the loss of generality, the model structure of this study can be formulated as:

$$\boldsymbol{y}_i = X_i \boldsymbol{\beta}_i + \boldsymbol{\epsilon}_i. \quad (1)$$

Moreover, the multi-task learning model induces generalization by forcing similarity in the model coefficients of related tasks. This induces the required knowledge transfer [27]. To enforce similarity in model coefficients, the model is formulated by penalizing for similarity and sparsity as:

$$\underset{\boldsymbol{B}}{\operatorname{argmin}} \sum_{i=1}^{t} L(\boldsymbol{\beta}_i, \boldsymbol{\epsilon}_i | X_i, \boldsymbol{y}_i) + \lambda_1 \omega(\boldsymbol{B}) + \lambda_2 \|\boldsymbol{B}\|_F^2, \quad (2)$$

where $\boldsymbol{B} = [\boldsymbol{\beta}_1^{\mathrm{T}}, \ldots, \boldsymbol{\beta}_t^{\mathrm{T}}]$ is the overall model coefficient matrix; $\omega(\boldsymbol{B}) = \sum_{i=1}^{p} \sqrt{\sum_{j=1}^{t} \boldsymbol{\beta}_{i,j}^2}$; and $\|\boldsymbol{B}\|_F^2 = \sum_{i=1}^{t} \sum_{j=1}^{p} \beta_{i,j}^2$; λ_1 and λ_2 are tuning parameters for similarity and sparsity; and t is the total number of tasks. In the multi-task learning model the overall model coefficient matrix $\boldsymbol{B}$ captures the variations in the relationship between the run-time metrics and predictors among the Fog nodes. The residuals $\boldsymbol{\epsilon}_i$ reflect the variability of the run-time metrics within a Fog node. The first term in Model (2) is the least square function to minimize the squared error of the model. The second term is a $l_{2,1}$ norm that enforces knowledge transfer between tasks by encouraging similarly sparse structured model coefficients [30]. The $l_{2,1}$ norm encourages the selection of similar model coefficients for the Fog nodes, thereby reducing variation in the model prediction between the Fog nodes. The third term is a Frobenius norm that favors sparsity in the model coefficients [31] and reduces variation in the model prediction within a Fog node by limiting the number of non-zero model coefficients. The objective function is solved by the accelerated gradient descent method. It can effectively yield the global optimal in a timely manner than the traditional iterative shrinkage algorithms [32]. The tunning parameters are chosen based on a 5-fold cross-validation to minimize the objective function using the training dataset [29].

CHAPTER 4: CASE STUDY

4.1 Experimental setup

To validate the proposed model, an existing Fog manufacturing testbed that can effectively generate manufacturing data and execute simple predictive data analytics pipelines is employed [18]. The multi-task learning model is evaluated by comparing the prediction accuracy for the run-time metrics with other benchmark methodologies (i.e., linear regression [40], Lasso regression [40], and Random forest regression [41]). These benchmarks are chosen, because previous work by Wang, L., et al. showed that Lasso and Random forest regression can competently predict performance in a Fog manufacturing setup [22,32]. Moreover, the variable selection results are validated to identify whether the proposed model can properly capture the heterogeneous property among the Fog nodes.

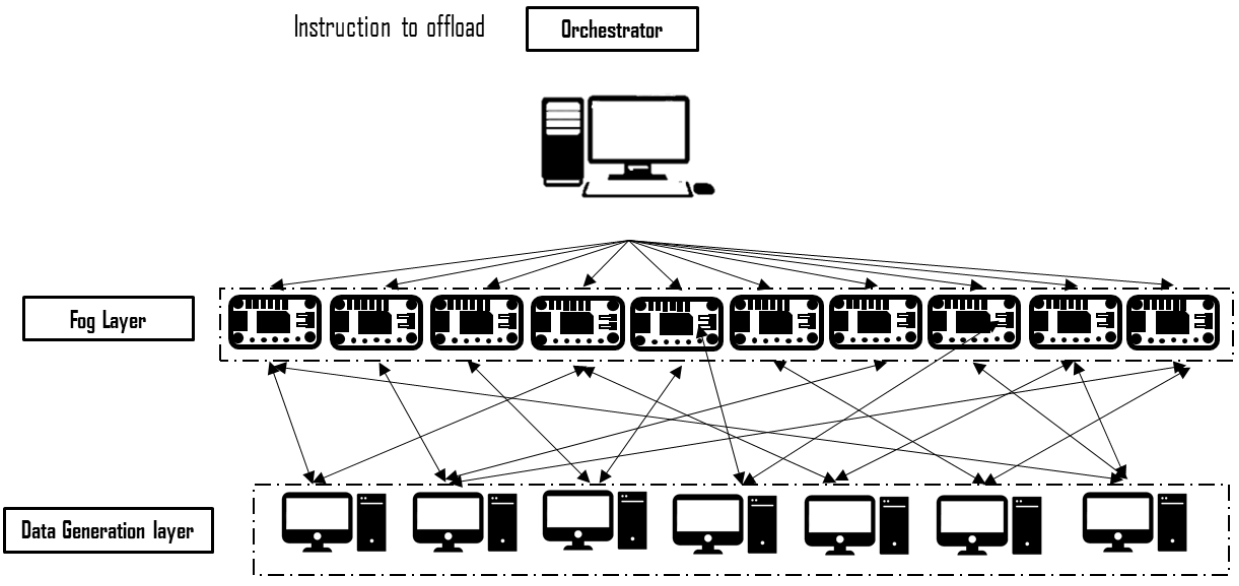

Figure 4: Fog manufacturing testbed (redrawn from [20])

In this case study, simple predictive data analytics pipelines generated from the data generation layer are offloaded to the Fog manufacturing testbed. The architecture of the testbed is shown in Figure 2. It can be observed that this testbed consists of three layers. Starting from the data generation layer, seven manufacturing datasets (i.e., a set of predictor and response variables) based on a simulated plasma chemical vapor decomposition (PCVD) process [35] are generated. The generated data are then taken through a computational flow where the simple predictive data analytics pipelines are broken into sub steps (i.e., feature extraction, parameter tunning and model training). These sub-steps are further offloaded to Fog nodes at the second layer. The second layer consists of 10 Fog nodes (i.e., Raspberry Pi 3 units) that have the similar computation and communication features. The Fog nodes are capable for storing, processing, and transmitting the data in this testbed. The final layer is the orchestrator, which is a workstation (CPU i7-6700k), is responsible for implementing the offloading and data transmission decisions in the Fog nodes.

To properly implement the simple predictive data analytics pipelines in Fog manufacturing, the orchestrator ranks the pipelines based on AdaPipe pipeline selection strategy [34]. The strategy prioritizes pipelines based on their statistical performance. Once the pipelines are identified and selected for offloading, the orchestrator decides upon assigning sub steps to the Fog nodes to maximize the number of completed pipelines in given time. The data sets for the corresponding sub steps are offloaded to the Fog nodes via a virtual offloading platform.[39]. The pipeline selection and offloading principles are extended to various combination of offloading scenarios shown in Table 2.

Table 2: Design of Experiments Factors

Factor	**Level 1**	**Level 2**
Pipeline selection	Random selection	Recommendation using AdaPipe selection strategy [34]
Pipeline number	Top 5 Pipelines	Top 10 Pipelines
Offloading assignment	Random assignment	Time balanced offloading
Data storage	One copy on each node	Three copies on three random nodes

To achieve different offloading scenarios in Fog manufacturing and comprehensively validate the multi-task learning model, a full factorial design of experiments with four factors (two levels for each) was employed. The design table is shown in Table 2. From Zhang et al., a comprehensive analysis on the factors dictating performance in Fog manufacturing showcased that the strategy of computation selection, number of computational flows, data storage, and offloading strategies effect performance in Fog computing. Hence these factors are chosen to cover the spectrum of offloading scenarios.

Specifically, pipeline selection refers to a method of determining simple predictive data analytics pipelines to be executed in the experiment. The AdaPipe is a pipeline selection strategy [34] that can prioritize the simple predictive data analytics pipelines based on their statistical performance for a specific dataset. Pipeline number is defined as the number of pipelines in total that will be executed. The offloading assignment strategy determines how the orchestrator will offload the sub-steps in selected pipelines to specific Fog nodes. Time balanced offloading means the orchestrator will assign sub-steps to the Fog nodes by maximizing the number of completed pipelines in given time [39]. Finally, the data storage strategy determines how the data will be allocated among the

Fog nodes. This characteristic can directly influence the data query efforts when executing the computation tasks.

It is worth mentioning that there exist alternative offloading strategies (e.g., offloading to reduce cost [17], offloading to minimize power consumption with latency constraints [19], etc.), alternate pipeline selection, and data storage strategies which are not accounted for in the current experimental design. The experimental testbed does not select these alternate scenarios and therefore the current model cannot be used to predict the run-time performance of the Fog nodes when new pipelines are deployed under different offloading and data querying scenarios. Since data for these alternate scenarios cannot be interpolated from the experimental data set. The scope of the current multi-task learning model is restricted to the levels explored in the experimental design. The robustness and reliability of the multi-task learning model can be improved for other identified levels of offloading treatments in the future work.

Table 3: Description of the Predictor Variables

Type	**Dummy/Continuous variables**	**Description**	**Variable index**
In situ variables (historical and last step)	Continuous	Latency, temperature, time stamp, power consumption, memory, CPU utilization, bandwidth utilization (mean, std, kurtosis, and skewness)	1-65 (Historical and Last step)
Setting variables (last step and current step)	Continuous	Pipeline description	78-90 (Last step) 107-116

			(Current step)
Setting variables (last step and current step)	Dummy	Step in pipeline executed	91 – 93 (Last step) 117 – 1119 (Current step)
Setting variable (last step and current step)	Continuous	Information on data processed, predictor and response variable (number of observations, mean, std, kurtosis, and skewness of the response and predictor variables)	66-76 (Last step) 95-105 (Current step)

To collect the necessary data from the testbed, when a sub-step is completed on a Fog node, important data, such as historical *in situ* variables on the Fog node's performance, pipeline description, and sub-step in the pipeline implemented are stored in a local database. The sample size for each Fog nodes reflect the total sub-steps completed in the Fog node; In the testbed, a total of 3,407 sub-steps were completed by the 10 Fog nodes with each Fog node completing between 290 to 371 sub-steps. The maximum number of offloaded sub-steps to a Fog node being 371 and the minimum number of offloaded sub-steps to a Fog node being 290. In the current experimental design, the Fog nodes have relatively balanced sample sizes (i.e., the number sub steps completed in the Fog nodes are comparable). However, if there exists a high imbalance in the sample sizes among the Fog nodes then the least square function term in the model formulation would be

influenced by the mean squared error of the Fog nodes with larger sample sizes. This affects the model estimation and variable selection effort.

A full list of the variable collected during the process is shown in Table 3. It is worth to mentioned that the description of the pipeline shown in Table 3 are continuous variables that are generated based on the pipeline description. This is achieved by using a Non-Negative matrix decomposition method [36] where a combination of unique features can adequately describe the characteristics of the pipelines.

4.2 Histogram of Run-time Metrics

The run-time metrics that are predicted by the multi-task learning model are Time- latency, CPU utilization, and Download utilization.

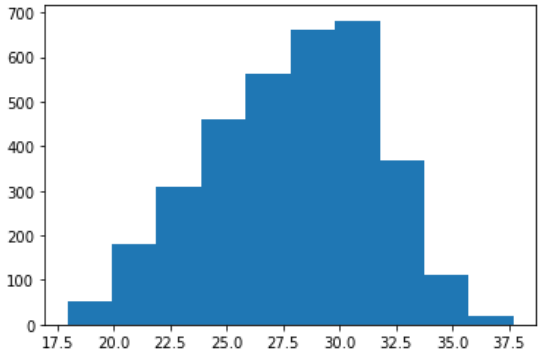

Figure 5: Distribution of CPU Utilization

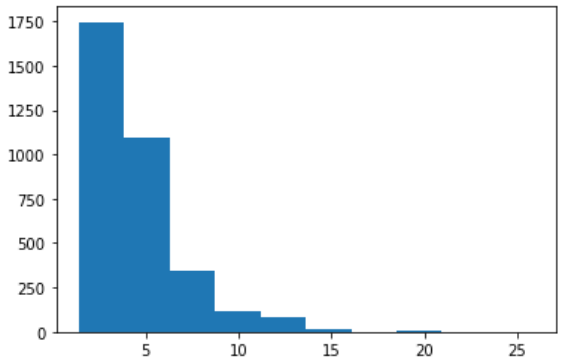

Figure 6: Distribution of Time-latency

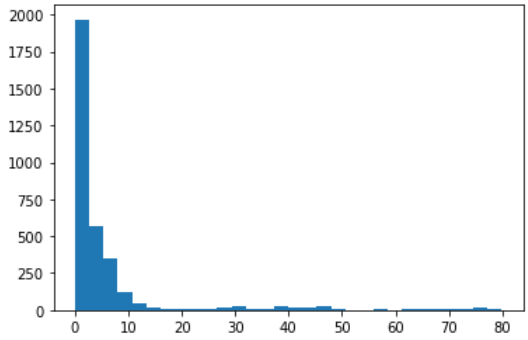

Figure 7: Distribution of Download utilization

The above histograms depict the distribution of the response variables CPU utilization, Time-latency. It is observed that Time-latency and Download utilization are skewed, a logarithmic operation is implemented to normalize the response variables [37]. In the case study, even though CPU utilization is truncated, the variance is relatively small, and a large sample of data is collected between the lower and upper bound. The histogram plot for CPU utilization reflects this. Hence the current multi-task learning model is sufficient in providing an unbiased estimate for predicting the CPU utilization. This is reflected by the absence of heteroscedastic in the diagnosis. If the variance of the truncated response variable is large, the maximum likelihood estimator for truncated data [42] can be used instead of the least square estimate to get an unbiased model

coefficient. Alternatively, the truncated data can be normalized through the means of logarithmic operations [37].

To standardize the prediction accuracy and compare it with existing benchmark methods, a normalized root-means-squared-error (NRMSE) method is utilized. The normalized root mean squared error is defined as: $\mathrm{NRMSE} = \frac{\sqrt{\sum_{i=1}^{N} \frac{(y_i - \widehat{y_i})^2}{N}}}{(y_{max} - y_{min})}$.

CHAPTER 5: RESULTS

5.1 Prediction and Variable Selection Results

The NRMSE for the proposed methodology and benchmark methods are shown in Table 4. It can be observed that the multi-task learning model outperforms linear regression, Lasso regression, and Random forest. This is because multi-task learning can model the data heterogeneity presented among the different Fog nodes by inducing a common meaningful bias without increasing the variance.

Table 4: NRMSEs for the Predictive Models

Model	Normalized Time-latency	CPU utilization	Log Download utilization
Linear Regression	0.131	0.081	0.165
Lasso	0.127	0.072	0.163
Random Forest	0.098	0.056	0.156
Multi-task Learning	**0.070**	**0.030**	**0.103**

The average of NRMSEs for time-latency, CPU utilization, and Download utilization are 0.070, 0.030, and 0.103, respectively. Comparing the accuracy with linear, Lasso, and random forest regression, it can be observed that multi-task learning can adequately identify key predictor variables better than the existing benchmark methods. This can be explained as multi-task learning models the heterogenous nature between the Fog nodes and transfers meaningful knowledge by forcing the model coefficients of the Fog nodes to be similar. When looking at the benchmark methods, random forest has better prediction accuracy than linear regression and Lasso regression model. It is because the random forest is an ensemble of decision trees that split till adequate depth

is achieved. This reduces the variance without increasing the model bias [41]. Lasso has slightly better prediction accuracy as the model not only aims at reducing the least square error, but also penalizes for sparsity. However, the linear regression model only aims at reducing the least square error and has the worst prediction accuracy among the benchmark models.

To investigate the model coefficients of the Fog nodes, a heatmap on the absolute values of model coefficients are shown in Figures 5, 6, and 7. By comparing the model coefficients, we validate the initial assumption on the heterogenous nature among the Fog nodes.

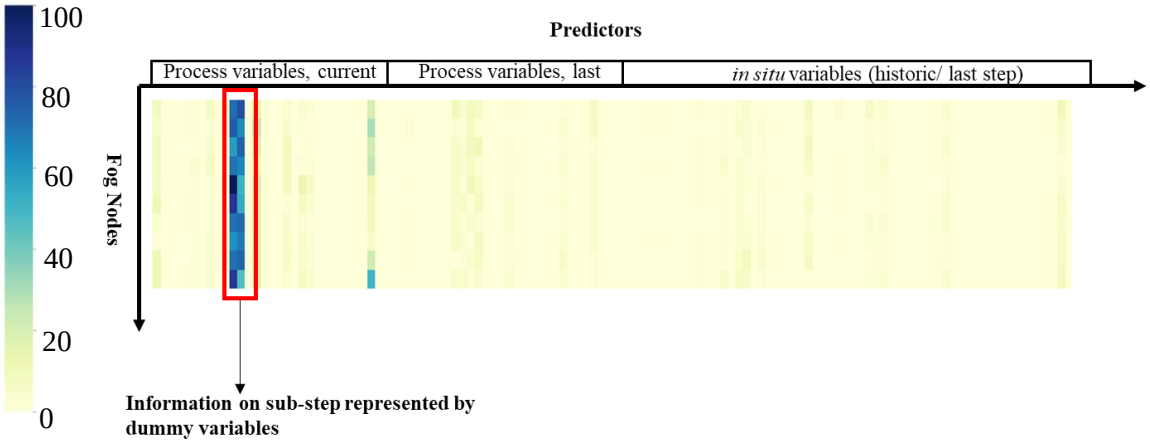

Figure 8: Magnitude of the Model Coefficients for Time-Latency

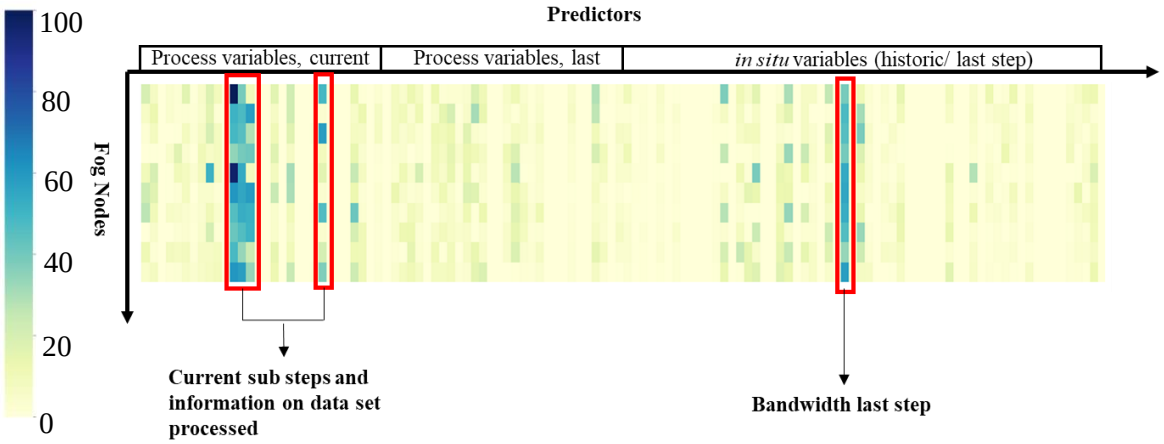

Figure 9: Magnitude of the Model Coefficients for Download Utilization

Specifically, a comparison of the estimated model coefficients among Fog nodes for Time-latency and Download bandwidth utilization are shown in Figure 5 and Figure 6. The colors in the heatmaps show the absolute values of the model coefficients scaled from zero to a hundred. The darker the color, the larger the magnitude of the model coefficients. The x-axis represents the predictors listed in Table 3 and the y-axis represents the different Fog nodes. From the figures, it can be observed that there is a similarity in pattern of model coefficients among the Fog nodes. Hence, the initial assumption that similarity in the Fog nodes exists. The multi-task learning model captures this effect through the variable selection process is validated. Especially, in Figure 5, it can be found that the current sub-step has the most significant impact when predicting the time-latency. It is because the current step can directly determine the computational workload for this sub-step and then influence the completion time of this task.

From Figure 6, It can be observed that the significant model coefficients for Download utilization prediction are the current steps, information on the data processed, and the accumulative effects from the previous steps. One possible explanation is that Download utilization depends on the data

query strategy and the size of the dataset from historical computation steps. The data transmission efforts from the previous tasks might have an accumulative effect to the bandwidth utilization.

To clearly demonstrate the accumulative effect on the Fog node, magnitude of the model coefficients for CPU utilization is shown in Figure. 7.

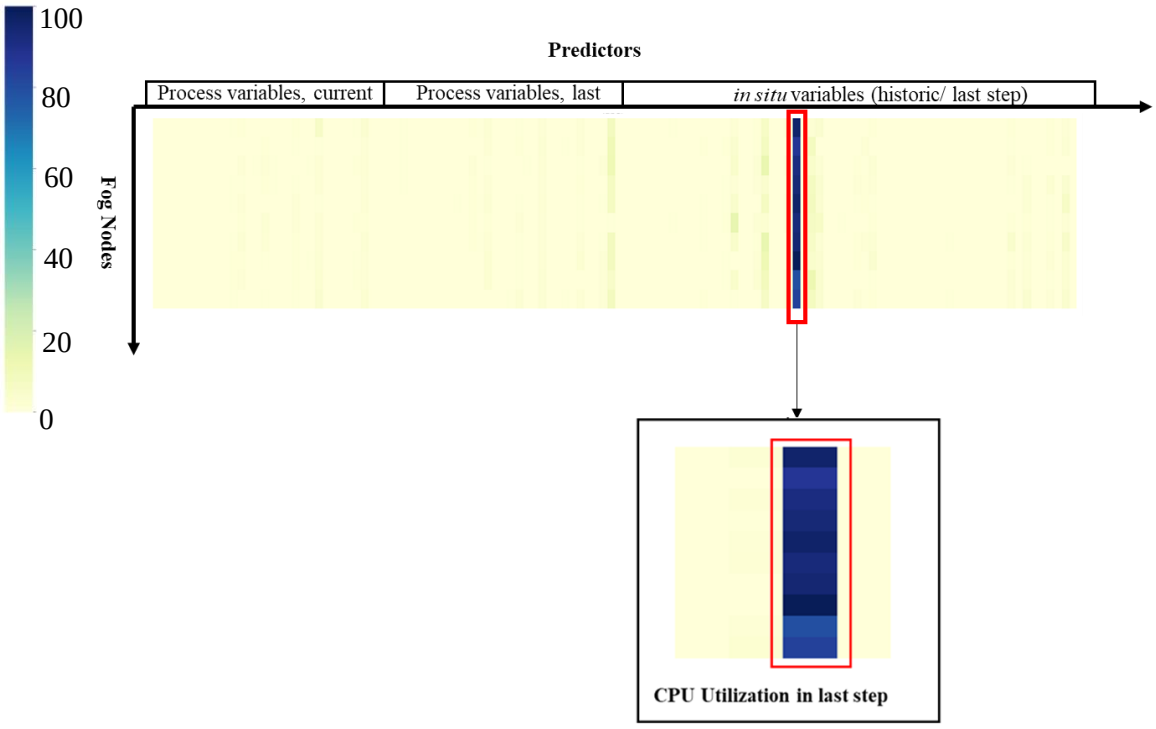

Figure 10: Magnitude of the Model Coefficients for CPU Utilization

From Figure. 7, it can be observed that the CPU utilization in the current step is highly related to the CPU utilization in the previous step. One possible explanation for the background computing tasks is that the previous computation can carry over to the next computational step. These computation workloads can further result in cumulative effects to the CPU utilization in practice.

A zoomed in view on the model coefficients comparison between the Fog nodes shows that the values of the model coefficient differ between the Fog nodes. To test if the difference is significant,

a hypothesis test to compare the two-regression model was conducted [36]. For example, a hypothesis test shows the model coefficients for Fog node 1 and Fog node 9 differ, which also explains why the multi-task learning model is better than the linear regression and Lasso. This is primarily due to varying cumulative computation workloads that differentiates the Fog nodes over time.

5.2 Diagnostics and assumption validation

To validate the initial assumption made when adapting multi-task learning, the diagnostic plots for Time-latency, CPU utilization, and Download utilization are shown below:

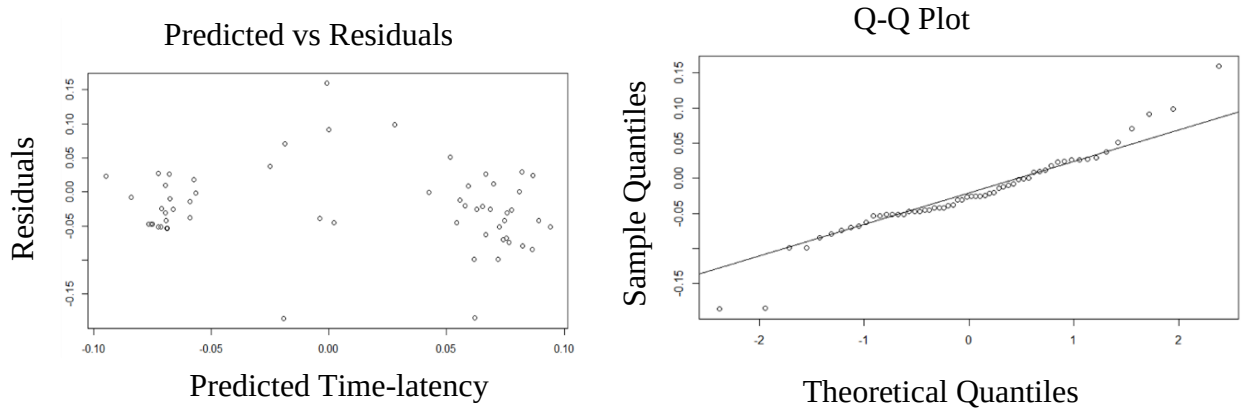

Figure 11: Diagnostic plots for Time-latency

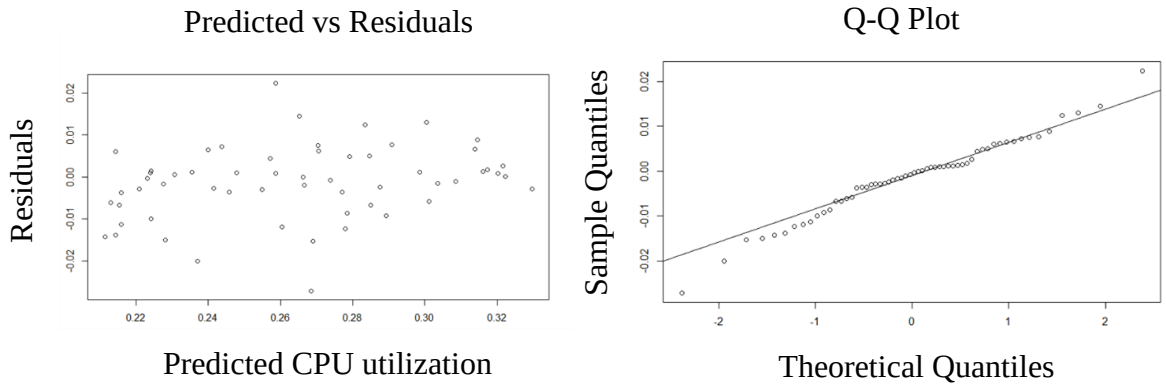

Figure 12: Diagnostic plots for CPU utilization

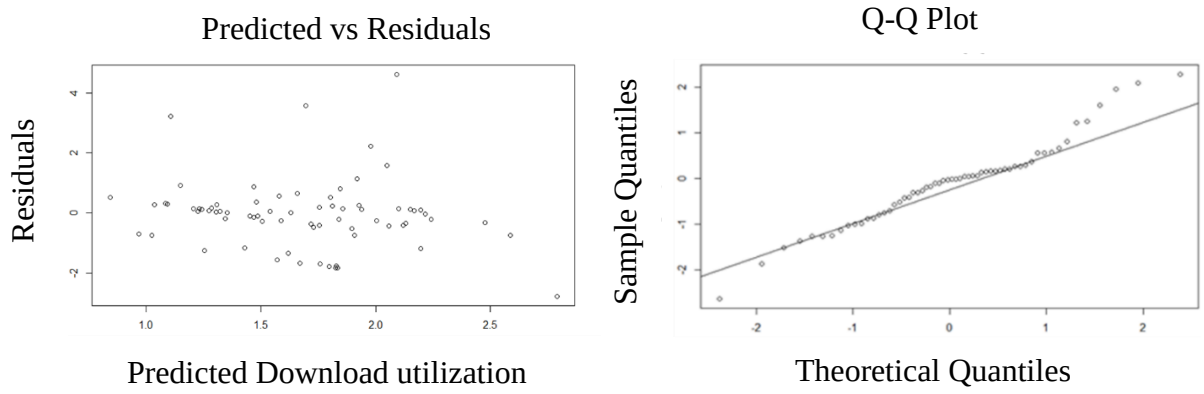

Figure 13: Diagnostic plots for Download utilization

From the diagnostic plots, it can be observed that the predicted v.s. residuals plots show no form of heteroskedasticity and the residuals follow a normal distribution. The prediction accuracy and diagnostics of the model validate the initial assumption that similarities in the run-time metrics among the Fog nodes exists. Also, a linear relationship is valid to explain the variance between the response and predictor variables.

CHAPTER 6: CONCLUSION

In smart manufacturing, providing a reliable and responsive computation service becomes critical in facilitating real-time decision making and avoiding catastrophic failures. Fog manufacturing is an emerging concept where data can be processed locally through a network of geographically distributed Fog nodes, without the need to transfer it directly to the Cloud servers for analysis. This helps in completing tasks that are urgent with less latency and decreases workload in the cloud servers. To effectively utilize Fog manufacturing, optimal computation task offloading strategies are necessary. Therefore, it is necessary to accurately predict the run-time metrics on Fog nodes to support the offloading efforts. However, current predictive models either extrapolate based on historic data, or do not consider heterogeneity in the run-time metrics among the Fog nodes.

To overcome this challenge, a regularized multi-task learning approach is adopted in this study. In multi-task learning, related tasks are simultaneously modeled together to induce a common bias that can capture the required domain knowledge. This solves the challenge of data heterogeneity among the Fog nodes. The methodology was validated using a Fog manufacturing testbed, where sub-steps in simple predictive data analytics pipelines were offloaded to ten Fog nodes. The proposed multi-task learning methodology was used to predict Time-latency, CPU utilization, and Download bandwidth utilization of the Fog nodes. The NRMSEs were 0.07, 0.03, and 0.10 respectively, which were better than linear regression, Lasso regression, and Random forest. The variable selection results were also presented to validate the assumptions made. The most important contribution of this paper is that the proposed model solves the problem of data

heterogeneity among the Fog nodes and predicts the performance run-time metrics of the Fog nodes before the implementation of a computational task.

CHAPTER 7: FUTURE WORK

In this thesis, the Fog nodes have similar computing and communicating capabilities. However, this might not be a feasible scenario in a real manufacturing environment. One future work direction for this study is to extend the multi-task learning in a Fog manufacturing testbed that has different configurations of Fog nodes. On the other hand, an immediate application of the multi-task learning model proposed in this study is facilitating the real-time predictive offloading in Fog manufacturing testbed. Moreover, the proposed multi-task learning model can also be extended to predictive autoscaling methods in Fog computing platform to dynamically optimize the architecture design.

REFERENCES

1. Wu, D., et al. *Fog-enabled architecture for data-driven cyber-manufacturing systems*. in *International Manufacturing Science and Engineering Conference*. 2016. American Society of Mechanical Engineers.
2. Mocanu, S., et al. *Fog-based solution for real-time monitoring and data processing in manufacturing*. in *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)*. 2018. IEEE.
3. Jain, A. and P. Singhal. *Fog computing: Driving force behind the emergence of edge computing*. in *2016 International Conference System Modeling & Advancement in Research Trends (SMART)*. 2016. IEEE.
4. Bouzarkouna, I., et al. *Challenges facing the industrial implementation of og computing*. in *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*. 2018. IEEE.
5. Chen, X. and R. Jin. *Data Fusion Pipelines for Autonomous Smart Manufacturing*. in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. 2018. IEEE.
6. Sparks, E.R., et al. *Keystoneml: Optimizing pipelines for large-scale advanced analytics*. in *2017 IEEE 33rd international conference on data engineering (ICDE)*. 2017. IEEE.
7. Bonomi, F., et al. *Fog computing and its role in the internet of things*. in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. 2012.

8. Brzoza-Woch, R., et al. *Embedded systems in the application of fog computing—Levee monitoring use case*. in *2016 11th IEEE Symposium on Industrial Embedded Systems (SIES)*. 2016. IEEE.
9. Rahmani, A.M., et al., *Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach.* Future Generation Computer Systems, 2018. **78**: p. 641-658.
10. Qi, Q., et al. *Modeling of cyber-physical systems and digital twin based on edge computing, fog computing and cloud computing towards smart manufacturing*. in *ASME 2018 13th International Manufacturing Science and Engineering Conference*. 2018. American Society of Mechanical Engineers Digital Collection.
11. Verba, N., et al., *Modeling industry 4.0 based fog computing environments for application analysis and deployment.* Future Generation Computer Systems, 2019. **91**: p. 48-60.
12. Aazam, M., S. Zeadally, and K.A. Harras, *Deploying fog computing in industrial internet of things and industry 4.0.* IEEE Transactions on Industrial Informatics, 2018. **14**(10): p. 4674-4682.
13. Yin, L., J. Luo, and H. Luo, *Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing.* IEEE Transactions on Industrial Informatics, 2018. **14**(10): p. 4712-4721.
14. Jiang, Y.-L., et al., *Energy-efficient task offloading for time-sensitive applications in fog computing.* IEEE Systems Journal, 2018. **13**(3): p. 2930-2941.
15. Jamil, B., et al., *A job scheduling algorithm for delay and performance optimization in fog computing.* Concurrency and Computation: Practice and Experience, 2020. **32**(7): p. e5581.

16. Luong, D.-H., et al. *Predictive Autoscaling Orchestration for Cloud-native Telecom Microservices*. in *2018 IEEE 5G World Forum (5GWF)*. 2018. IEEE.
17. Casalicchio, E. and S. Iannucci, *The state-of-the-art in container technologies: Application, orchestration and security.* Concurrency and Computation: Practice and Experience, 2020: p. e5668.
18. Zhang, K., et al., *Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading.* IEEE Vehicular Technology Magazine, 2017. **12**(2): p. 36-44.
19. Gao, X., et al., *PORA: Predictive offloading and resource allocation in dynamic fog computing systems.* IEEE Internet of Things Journal, 2019. **7**(1): p. 72-87.
20. Zhang, Y., et al. *Fog Computing for Distributed Family Learning in Cyber-Manufacturing Modeling*. in *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*. 2019. IEEE.
21. Chen, X., et al. *Predictive offloading in mobile-fog-cloud enabled cyber-manufacturing systems*. in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. 2018. IEEE.
22. Patman, J., et al. *Predictive analytics for fog computing using machine learning and GENI*. in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2018. IEEE.
23. Clemente, J., et al. *Fog computing middleware for distributed cooperative data analytics*. in *2017 IEEE Fog World Congress (FWC)*. 2017. IEEE.
24. Baxter, J. *A Bayesian/information theoretic model of bias learning*. in *Proceedings of the ninth annual conference on Computational learning theory*. 1996.
25. Caruana, R., *Multi-task learning.* Machine learning, 1997. **28**(1): p. 41-75.

26. Huang, Y., J.L. Beck, and H. Li, *Multi-task sparse Bayesian learning with applications in structural health monitoring.* Computer-Aided Civil and Infrastructure Engineering, 2019. **34**(9): p. 732-754.

27. Li-na, C. and Z. Pei-ai, *Application of Multi-task Lasso Regression in the Parametrization of Stellar Spectra.* Chinese Astronomy and Astrophysics, 2015. **39**(3): p. 319-329.

28. Evgeniou, T. and M. Pontil. *Regularized multi--task learning*. in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004.

29. Cao, H., J. Zhou, and E. Schwarz, *RMTL: an R library for multi-task learning.* Bioinformatics, 2019. **35**(10): p. 1797-1798.

30. Liu, J., S. Ji, and J. Ye, *Multi-task feature learning via efficient l2, 1-norm minimization.* arXiv preprint arXiv:1205.2631, 2012.

31. Nie, F., et al. *Efficient and robust feature selection via joint ℓ2, 1-norms minimization*. in *Advances in neural information processing systems*. 2010.

32. Wang, L., et al. *Online Computation Performance Analysis for Distributed Machine Learning Pipelines in Fog Manufacturing*. in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. 2020. IEEE.

33. Beck, A. and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems.* SIAM journal on imaging sciences, 2009. **2**(1): p. 183-202.

34. Chen, X. and R. Jin, *AdaPipe: A Recommender System for Adaptive Computation Pipelines in Cyber-Manufacturing Computation Services.* IEEE Transactions on Industrial Informatics, 2020.

35. Bower, C., et al., *Nucleation and growth of carbon nanotubes by microwave plasma chemical vapor deposition.* Applied Physics Letters, 2000. **77**(17): p. 2767-2769.

36. Gupta, M.D. and J. Xiao. *Non-negative matrix factorization as a feature selection tool for maximum margin classifiers*. in *CVPR 2011*. 2011. IEEE.

37. Box, G.E. and D.R. Cox, *An analysis of transformations.* Journal of the Royal Statistical Society: Series B (Methodological), 1964. **26**(2): p. 211-243.

38. Bruin, J., *Newtest: command to compute new test.* UCLA: Statistical Consulting Group, 2006.

39. G. Pemmasani, "dispy: Distributed and Parallel Computing with/for Python," 2020. [Online]. Available: http://dispy.sourceforge.net/.

40. Hastie, T., R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. 2009: Springer Science & Business Media.

41. Breiman, L., *Random forests.* Machine learning, 2001. **45**(1): p. 5-32.

42. Orme, C.D. and P.A. Ruud, *On the uniqueness of the maximum likelihood estimator.* Economics Letters, 2002. **75**(2): p. 209-217.