Refining Optimization Methods for Training Machine Learning Models: A Case Study in Robotic Surgical Procedures

Francis Boabang

A Thesis

in

Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy(Information Systems Engineering)

at

Concordia University

Montréal, Québec, Canada

December 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Francis Boabang

Entitled: Refining Optimization Methods for Training Machine Learning Models: A Case Study in Robotic Surgical Procedures

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy(Information Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

Dr. Marius Paraschivoiu	Chair
Dr. Christian Gagné	External Examiner
Dr. Juergen Rilling	Arms-Length Examiner
Dr. Yang Wang	Examiner
Dr. Jamal Bentahar	Examiner
Dr. Farnoosh Naderkhani	Thesis Supervisor

Approved by

Dr. Farnoosh Naderkhani, Graduate Program Director
Concordia Insitute for Information Systems Engineering

Date of Defence: September 4, 2024

Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Refining Optimization Methods for Training Machine Learning Models: A Case Study in Robotic Surgical Procedures

Francis Boabang, Ph.D.

Concordia University, 2024

Machine learning is a technology that builds predictive models from data, allowing generalization to unseen cases. At the core of every learning problem lies an optimization challenge, and solving these problems reliably is crucial to resolving the obstacles surrounding machine learning. Primarily, conventional optimization algorithms employed for training machine learning frequently are often ill-suited for various applications. Concerted efforts are needed to refine and optimize various components of machine learning training. This thesis explores fundamental optimization algorithms across various machine learning applications. By enhancing optimization schemes, including optimizers and model compression techniques, the resilience and effectiveness of machine learning applications can be improved.

The first segment of the thesis introduces an innovative low-rank matrix factorization scheme aimed at enhancing the scalability of machine learning. Gaussian Process Regression is used as the machine learning model to scale with low rank matrix factorization in this section of the thesis due to its lightweight nature, which enables the incremental updating of model parameters online prior to prediction. A nonconvex formulation of a low-rank matrix factorization (SRLSMF) with convex formulation of a low-rank matrix factorization initialization (ℓ_1-SRLSMF), is advocated to scale Gaussian Process Regression (GPR). Thus, by employing convex nonconvex low rank matrix factorization to scale a given the Gaussian Process Regression model, the model can avoid local minima and converge to a solution with smaller recovery residuals. Also, the running time of convex nonconvex low rank matrix factorization is expected to be smaller than that of applying nonconvex low rank matrix factorization alone under the same stopping criterion. To the best of our knowledge, the machine learning method proposed in this thesis is the first to exploit nonconvex formulation of a low-rank matrix factorization (SRLSMF) with convex formulation of a low-rank matrix factorization initialization (ℓ_1-SRLSMF) to scale machine learning in machine learning domain. Recognizing the cost-prohibitive nature of standard eigen decomposition for online Gaussian Process Regression covariance update, we implement incremental eigen decomposition within the ℓ_1-SRLSMF and SRLSMF Gaussian

Process Regression methodologies. Finally, an illustration of the potential applications in suturing, knot-tying and needle passing task using kinematic dataset is provided.

In the latter part of the thesis, a novel adaptive stochastic gradient descent (ASGD) method, which leverages the non-uniform p-norm concept to train machine learning is presented. The proposed ASGD assigns distinct categories of coordinate values with varying base learning rates, thereby enabling the training of machine learning models. Additionally, theoretical guarantees for the efficacy of the proposed ASGD method in convex and nonconvex settings is discussed. The ability of the proposed ASGD approach to detect suturing gestures within the remote surgical gesture recognition task is discussed.

Finally, potential avenues for future research are outlined.

Acknowledgments

I would like to express appreciation for the contributions of my committee, collaborators, and administrative supervisors to the thesis. Special thanks go to Dr. Jun Yan and Dr. Farnoosh Naderkhani for their support and guidance throughout the thesis. I extend my heartfelt gratitude to Dr. Roch Glitho for his invaluable support, encouragement, and exceptional guidance throughout this endeavor. His vast knowledge and expertise have been instrumental in introducing the topic of remote robotic surgery and providing invaluable supervision. I am also appreciative of my committee members, namely Dr. Chadi Assi, Dr. Juergen Rilling, Dr. Jamal Bentahar, Dr. Manar Amayri, Dr. Wang Yang, external examiners Dr. Daniel Aloise and Dr. Christian Gagne for providing constructive feedback and insights that significantly enhanced my research. Additionally, gratitude extends to my collaborators, including Dr. Elbiaze Halima, Dr. Martin Maier, Dr. Wessam Ajib, Dr. Amin Ebrahimzadeh, Dr. Omar Alfandi, and Dr. Fatna Belqasmi, for their valuable insights and feedback on the thesis composition.

I extend my sincere gratitude to the department staff members, Ms. Silvie Pasquarelli and Ms. Mireille Wahba, for their invaluable support and assistance. Additionally, I would like to extend my gratitude to my friends, Dr. Kojo Sarfo Gyamfi and Mr. Ronald Ekow Crankson Ellis, for their encouragement.

I express profound appreciation to my mother, father, and grandmother for their unwavering encouragement and steadfast support throughout my academic journey. They stood by my side, offering guidance and encouragement every step of the way.

I am grateful to the anonymous reviewers for their invaluable feedback, which significantly contributed to enhancing the quality of the published and unpublished papers utilized for this thesis.

This thesis draws inspiration from the renowned 26-page thesis by Dr. John Nash, dated 1950.

List of published and unpublished papers

Journal Francis Boabang, A. Ebrahimzadeh, R. Glitho, H. Elbiaze, M. Maier and F. Belqami, ”A Machine Learning Framework for Handling Delayed/Lost Packets in Tactile Internet Remote Robotic Surgery,” in IEEE Transactions on Network and Service Management, doi: 10.1109/TNSM.2021.3106577.

Conference Francis Boabang, R. Glitho, H. Elbiaze, F. Belqami and O. Alfandi, ”A Framework for Predicting Haptic Feedback in Needle Insertion in 5G Remote Robotic Surgery,” 2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC), 2020

Journal Francis Boabang, ”Enhanced Stochastic Gradient Descent Algorithm for Machine Learning Training and Its Application in Remote Surgical Gesture Recognition,” to be submitted.

Contents

List of Figures

List of Tables

Table 1 List of notations and their meanings.

Symbol	**Definition**
B	Column matrix
X_{clean}	Clean kernel matrix
D_{knot}	Total number of knot-tying sessions
E	Difference between the kernel matrix G and approximated matrix L
F	The representation of column matrix B with U basis matrix
G	Kernel matrix
$G(\phi^{(d_{knot})}, \phi^{(d_{knot})})$	Covariance function
$G(\phi^{(d_{knot})}, \phi^{(d_{knot}*)})$	Covariance matrix between the full training points and test points
$G(\phi^{(d_{knot})}, \phi^{(d'_{knot})})$	Covariance (or Kernel) function evaluated at $\phi^{(d_{knot})}$ and $\phi^{(d'_{knot})}$
J	Total number of iterations
$L_{lowrank}$	Approximated kernel matrix
M	Total number of datapoints in a knot-tying session
P	Auxiliary variable
Q	Orthogonal matrix
S	Matrix of packet loss/delay
T_{cov}	Covariance matrix
U	Orthogonal matrix of the approximated kernel matrix $L_{lowrank}$
G_1	Concatenated kernel matrix G and column matrix B
I_r or I	Identity matrix
d_{knot}	Index of a knot tying session
h_{layers}	Number of layers
i	Index of a column matrix
j	Iterative step
k	Number of states
m	Index of a step of the knot tying session
m'	Index of another step of the knot-tying session
r	Parameter to control the rank of L
x	Number of local models
Σ	Diagonal matrix of the approximated kernel matrix $L_{lowrank}$
$\beth$	Incoherence of the low rank matrix
ϱ	Condition number of the low rank matrix
Σ	Diagonal matrix
$\hat{\Sigma}$	Auxiliary variable
ϑ	Regularized covariance matrix
ψ	Knot tying session data
l	Rank of subset kernel matrix
μ	Mean vector
Ω	Gesture of a knot-tying task
γ	Maximal acceptable packet loss/delay
ϕ	Query vector
ξ	Target vector
Λ	Lagrange multipliers

Table 2 List of key notations.

Symbol	**Definition**
b_c	Bias vector of cell state
b_i	Bias vector of input state
b_f	Bias vector of forget gate
b_o	Bias vector of output gate
c_i	Cell state
C	small constant
D_∞	Bounded Diameter
d	Feature vector
f_i	Forget gate
$f(H)$	Linear Function
g	Loss Function
H	Second Order Momentum Matrix
$\hat{H}$	Maximum Second Order Momentum Matrix
z	Gradient
i	Gesture index
i_i	Input state
o_i	Output gate
$\mathbf{x}_i$	Video features
y_i	gesture of the suturing task i
t	Iteration step of the weight update
T	Total number of iteration of the weight update
α_{min}	Minimum Base Learning Rate
α_{max}	Maximum Base Learning Rate
u	Auxiliary variable
β	Adaptive parameter
ϵ	Small constant
W	Weight of the LSTM model
W_c	Weight matrix of cell state
W_{INPUT}	Weight matrix of input
W_f	Weight matrix of forget gate
W_o	Weight matrix of output
β	Momentum adaptive parameter
n	First Order Momentum Matrix
β	Weight parameter
λ	A penalty parameter for balancing the low-rank and the reconstruction fidelity
σ_n^2	Noise variance
σ_s^2	Noise variance of the kernel function itself
$\|\|.\|\|_F$	Frobenius norm
$\|\|.\|\|_*$	Nuclear norm
$\|\|.\|\|_1$	ℓ_1-norm

Table 3 List of abbreviations.

Abbreviation	**Meaning**
5G	Fifth Generation
4G	Fourth Generation
ADAM	Adaptive Moment Estimation
ANN	Artificial Neural Network
AP	Access Point
ASGD	Adaptive Stochastic Gradient Descent
BS	Base Station
DRMF	Direct Robust Matrix Factorization
DeepRL	Deep Reinforcement Learning
ELM	Extreme Learning Machine
EM	Expectation Maximization
FNN	Feed-Forward Neural Network
FiWi	Networks Fiber-Wireless Networks
GMM	Gaussian Mixture Model
GMR	Gaussian Mixture Regression
GPR	Gaussian Process Regression
HMM	Hidden Markov Model
JIGSAW	JHU-ISI Gesture and Skill Assessment Working Set
KNNR	K-Nearest Neighbor Regression
LDS	Linear dynamic system
LFD	Learning from demonstration
LR	Linear Regression
LSTM	Long Short Term Memory
MSE	Mean Square Error
MEC	Multi Access Edge Computing
MC	Mission Critical
MPP	Mesh Portal Point
NN	Neural Network
ONU	Optical Network Unit
PADAM	Partially Adaptive Momentum Estimation Method
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RL	Reinforcement Learning
SVD	Single Value Decomposition
RPCA	Robust Principal Component Analysis
LWPR	Locally Weighted Projection Regression
WLAN	Wireless Local Area Network

Chapter 1

Introduction

1.1 Overview

Machine learning has rapidly expanded, drawing a diverse group of researchers and practitioners. It has emerged as a leading research area, significantly impacting various domains, including machine translation, speech recognition, image recognition, and recommendation systems. A fundamental aspect of machine learning is optimization, which involves constructing models to learn parameters from data. To enhance machine learning development, numerous effective optimization methods have been introduced, boosting the performance and efficiency of these techniques. Common optimization approaches can be categorized into three areas: loss functions, model compression, and stochastic gradient descent.

Low-rank matrix factorizations have also gained popularity for projecting high-dimensional data into lower-dimensional latent spaces, improving data understanding and prediction accuracy. These methods are increasingly utilized to scale machine learning applications. Most existing low-rank matrix factorization approaches for scaling machine learning either rely on convex or nonconvex formulations. The convex formulation serves as an approximation of the nonconvex formulation, often resulting in suboptimal solutions for scaling machine learning. Conversely, the nonconvex formulation requires good initial points to achieve optimal results. Furthermore, most methods for scaling machine learning are non-incremental, meaning they struggle to incorporate new data or information in real time. This limitation can hinder their ability to enhance model generalization performance as fresh data becomes available.

Recently, adaptive stochastic gradient descent methods and their variants have gained traction, evolving quickly. However, many researchers treat these methods as black box optimizers, often neglecting their specific characteristics and application contexts. This can lead to suboptimal convergence rates, limiting their effectiveness.

In this thesis, advanced optimization schemes to optimize machine learning methods to provide algorithmic solutions for machine learning applications is proposed. The first contribution is a low rank matrix factorization for machine learning training. Then, the result, a two-step convex nonconvex (CNC) sparse and low rank matrix factorization [5] scheme is exploited to scale Gaussian process regression. The resulting machine learning framework is used to conduct reactive predictions in remote robotic surgery. The benefit of using the convex nonconvex sparse and low-rank matrix factorization is that the model can converge to a quality solution even under random initialization, thus ensuring the stability of the many systems. Additionally, the proposed model employs incremental eigen decomposition to update the model parameters in real time, enhancing generalization performance. The outcome of this contribution was published in IEEE Transactions on Network and Service Management.

The second contribution is a novel adaptive stochastic gradient descent for training machine learning. By dynamically adjusting the learning rate during training based on the size of coordinate value of the surgical data, the proposed ASGD enables navigation through complex and noisy optimization landscapes. Moreover, the proposed ASGD's ability to optimize learning rates based on the size of the coordinate values contributes to expedited convergence of the machine learning algorithm, crucial for real-time decision-making. Faster convergence of the proposed ASGD facilitates prompt adaptation to changing environments. The combination of effective optimization, resilience to variability, accelerated convergence, and decreased sensitivity to hyperparameters ultimately enhances the performance of many systems. The outcome of this contribution is to be submitted.

1.2 Background Information

1.2.1 Machine Learning

Machine learning endows systems with the ability to actively learn and improve their behavior, without the need to be programmed. The learning process is generally performed by analyzing input data to detect patterns or regularities. Machine learning can be divided into three categories, namely supervised, unsupervised and semi-supervised learning. Supervised learning utilizes data labels to build the learning model. It finds a mapping between the input and output (labels) in the training data. The output is finite and discrete for classification and continuous for regression problems [6]. Classification finds discriminators to distinguish multiple classes in data, while regression finds the function that maps the input to the output [6]. Both are used for prediction of unseen data namely, testing data. Data is unlabeled in unsupervised paradigm and the machine learning methods capture pattern occurrences in data based on the similarity among data instances. Unsupervised learning can be used for tasks like Density Estimation (DE), clustering, Dimensionality Reduction/Feature Extraction (DR/FE), and sample generation [7]. DE estimates the distribution of data occurrence. Clustering partitions data into groups, such that the data inside a group are similar to each other, while groups are different from each other [6]. DR/FE maps the data from the current space to a new space with lower dimension. In semi-supervised learning methods, both labeled and unlabeled data are used to improve the learning process [8]. However, in many machine learning applications, sufficient labeled data are available for training the model.

Shallow machine learning methods are also either supervised or unsupervised. Shallow learning methods is where the entire learning process is organized in a single layer of processing without any hierarchical or step-wise processing on data [9]. An example of a shallow learning model is Gaussian Process Regression. Gaussian Process Regression addresses the challenges of high dimensionality by approximating the function to be learned [10]. It uses a set of Gaussian, or normal, distributions to describe and approximate the function, fitting the approximation globally across the entire data space. In contrast, Locally Weighted Projection Regression (LWPR) decomposes the high-dimensional function into a set of single-dimensional functions fitted on spatially localized data spaces. This is done by projecting the input data space along a few selected dimensions, thereby reducing the number of dimensions. Each projection is then used to approximate the local function in the neighborhood of the queried data point. Gaussian Process Regression is known for achieving high performance with minimal parameter tuning, but it suffers from significantly high computational complexity.

Deep learning performs learning that can be used for both supervised and unsupervised learning paradigms. It processes the input in several layers such that the output of each layer is injected as input to the subsequent layer, with the final result at the last layer [9]. An example of a deep learning network approach is a feedforward neural network (FNN), where data flows from input to output units without any feedback connections between them [11]. Training FNNs is slow due to the gradient-based learning algorithms they use and the iterative nature of the training process. To accelerate FNN training, Extreme Learning Machine (ELM) methods propose a mechanism with a single hidden layer. In ELM, hidden nodes are randomly selected, and the output weights are determined analytically [11]. ELM methods exhibit extremely fast learning speeds and require less memory, but they are outperformed by recurrent neural networks (RNNs) in terms of generalization accuracy. RNNs are designed for time-series problems where predictions depend on both current and previous observations. They are trained using Back Propagation Through Time (BPTT) [12]. RNNs can be considered a form of deep learning since they can be viewed as several nonlinear layers of neurons when unfolded in time [12]. Standard RNNs suffer from the vanishing gradient problem, which hampers the learning of long data sequences, such as those required for knot-tying tasks [12]. Long Short-Term Memory (LSTM) networks address the vanishing gradient problem by using special linear memory cells that maintain their activation over extended periods. LSTMs produce more accurate and robust predictions than RNN methods, even with a high number of hidden layers. However, as the number of hidden layers increases, LSTM networks operate more slowly and with higher computational costs. Convolutions Neural Networks (CNNs) are another type of deep learning model that has become prevalent in computer vision applications and has gained significant attention in remote robotic surgery. CNNs consist of several neural network blocks, pooling layers, and fully connected layers. They are designed to automatically and adaptively learn spatial hierarchies of features through a backpropagation algorithm. CNNs have been used to design much deeper models such as WideResNet [13], ResNet [13], and VGGNet [14]. ResNet and VGGNet are designed by stacking CNN layers to increase depth, with VGGNet being much deeper than ResNet. WideResNet, on the other hand, focuses on using broader CNN layers to increase width, aiming to reducing the computational complexity of ResNet without sacrificing accuracy. Recently, deep learning methods have been combined with reinforcement learning (RL) to form Deep-reinforcement learning, which is required in highly dimensional environment where standard reinforcement learning does not apply.

Offline training eliminates the time required to update the model parameters online [15]. These strategies are generally not well suited for bilateral teleoperations where commands and sensor data are received in sequential streams and model parameters must be updated on-the-fly. Offline training may cause significant inaccuracies in prediction for a multitude of reasons [15]. Notably, given that it is not possible to learn the complete state space beforehand, models that have learned offline can only approximate the model correctly in the area of the state space that is covered by the sample; the trained model would not be able to generalize beyond that region. In contrast, online learning creates a more general model for a larger space, as it is capable of adapting for time-dependent virtual or real object dynamics. Online model learning is known to be computationally expensive. Locally Weighted Projection Regression (LWPR) is a widely used method to manage the computational burden associated with online model learning [15]. LWPR partitions the state space into local regions, approximating local models that do not account for the global behavior of the embedded functions. The advantages of robust locally weighted projection methods can be combined with well-established regression techniques such as support vector regression and Gaussian process regression [10]. This hybrid approach would enable real-time learning of a robot's inverse dynamics. However, a major drawback of LWPR is the need for manual parameter tuning to achieve optimal performance.

In robotics and machine learning, continuous data streams cause the training dataset to grow indefinitely. To adhere to memory and computational constraints, many approaches limit the number of training samples using a fixed buffer size [16]. As new samples arrive, older samples must be discarded to maintain the buffer size. Typically, the oldest samples are discarded based on the assumption that they are the most outdated [16]. This method, however, can result in the model losing important regression information. To address this issue, a framework is needed to effectively incorporate previously trained model information into subsequent updates without retraining on the entire dataset from scratch.

Optimization is a fundamental and essential tool for training machine learning applications. Convex optimization is a significant subclass of optimization problems where both the objective function and the constraints are convex [17]. While many real-world applications may not be inherently convex, reformulating or approximating these problems as convex programs, known as convex relaxation has been a common strategy, though it often yields suboptimal solutions [5,17]. In addition, relying on these relaxations and convex solvers can lead to reduced performance, as they may compromise solution quality. In contrast, non-convex solvers frequently outperform their convex counterparts in terms of both solution quality and convergence speed. This advantage arises from non-convex methods' ability to effectively leverage the underlying structure of the problem, which is often obscured in convex reformulations. As a result, the development of non-convex solvers has become a compelling research focus. Numerous algorithms have been proposed to address non-convex constrained optimization problems. Although these algorithms are generally applicable and easy to implement, their convergence can be excessively slow for many machine learning applications [17] leading to low solution quality. Therefore, there is an urgent need to investigate machine learning performance in the context of non-convex programs. This thesis presents an alternative optimization method aimed at overcoming these limitations, particularly regarding solution quality.

Central to every supervised, semi-supervised, online, and offline learning process are optimization techniques such as model compression and loss functions which ensure the machine learning algorithm performs well. Moreover, for machine learning to be effective in applications, achieving good generalization performance is crucial. Good generalization performance means that the machine learning model performs well on both training and testing data. Model compression, for example, uses regularization strategies to impose a penalty on the model's complexity and smoothness, promoting good generalization on unseen haptic data even when trained on a limited dataset. Regularization helps prevent overfitting.

Another optimization strategy that enhances machine learning performance for machine learning applications is adaptive stochastic gradient descent [18]. ASGD helps machine learning models escape local minima and converge to a good solution with sufficient iterations. It starts by initializing basic parameters such as momentum, loss function, and step-size. The descent direction is determined by these parameters. If the loss function is convex, the function reaches a good solution with fewer iterations compared to a non-convex loss function. The loss function measures the discrepancy between system observations and the output of the prediction function. SGD and model compression can work together to enhance machine learning performance; for instance, SGD can serve as the training algorithm for model compression techniques.

In the upcoming subsections, the challenges posed by current optimization schemes in training supervised learning models are discussed, then key contributions in addressing these issues are highlighted. Lastly, an outline of the thesis structure is presented.

1.2.2 Challenges

Non-convex optimization algorithms are essential in modern machine learning and deep learning. This raises the question: how can an optimization algorithm be systematically designed, analyzed, and interpreted? This thesis aims to provide a robust theoretical framework for non-convex optimization, developing and analyzing algorithms specifically for training machine learning models across various applications.

The first challenge of the thesis focuses on developing mathematical techniques to analyze non-convex low-rank matrix factorization optimization algorithms for various machine learning problems. A family of functions with favorable landscape properties that facilitate efficient optimization are explored, demonstrating that the objectives of several machine learning problems fall within this family. There are two main challenges for analyzing and designing non-convex optimization algorithms: a) finding a coarse approximate solution first, followed by local improvement algorithms, and b) applying local improvement algorithms from any type of initialization. The first paradigm allows for the use of precise mathematical tools and often yields strong theoretical guarantees, while the second is more practical, applicable to problems where coarse solutions are difficult to determine without local improvements. This thesis aim to address both challenge: first, by providing effective initialization to reach good solutions, and second, by allowing for incremental updates to model parameters online using incremental eigen decomposition. This enables coarse solutions to effectively enter the basin of attraction of some global minimum of the non-convex optimization landscape. Gaussian Process Regression is selected as the underlying model due to its lightweight nature and ease of updating the model parameter updates on-the-fly prior to inference.

Stochastic Gradient Descent (SGD) algorithms are widely used in many stochastic optimization problems, including machine learning. Despite their empirical success, the theoretical understanding of SGD convergence properties remains limited, particularly due to the challenging non-convex optimization landscape. Classical machine learning methods often lead to convex and smooth optimization problems, for which gradient descent can efficiently find solutions. In contrast, modern methods, such as deep neural networks, frequently involve non-smooth and non-convex problems that cannot be solved efficiently in theory. The second challenge of the thesis is how to find the effective performance of Adaptive Stochastic Gradient Descent algorithms through theoretical analysis and empirical results, utilizing a base learning rate adaptation strategy to be able to escape local optima and converge to good solution.

Key Challenges

1. One major concern with non-convex functions is the presence of poor local minima. In many cases, these functions contain multiple local minima that are not global minimum, and if they fail to meet specific regularity conditions, it can be challenging to optimize them effectively. To identify a local minimum that is also a global minimum, local improvement algorithms are often required.

2. The inability of a model to adapt to new data and incorporate new classes on the fly can significantly impact its generalization performance. Addressing this challenge is crucial to ensure that the model can integrate and adapt to new information without losing previously acquired knowledge. An efficient online incremental model should be lightweight and maintain high generalization performance to support real-time applications. It would be insightful to increase the number of instances and categories to observe how overall accuracy evolves, specifically, whether it stabilizes or declines below the current level. The challenge of incremental learning is dealt with by developing models that can adapt to sequential (non-stationary) data, allowing them to adjust their parameters to reflect the current data distribution.

1.2.3 Overview of Machine Learning for Robotic Surgery

Robotic surgery improves precision, increases surgeon's dexterity, reduces tremor and improves ergonomic conditions [19, 20]. Patients also benefit since it involves less blood loss and pain, and therefore reduced trauma, and shorter recuperation time. Robotic surgery occurs in close proximity between a surgeon and a patient. The Da Vinci Surgical system (Intuitive Surgery Inc) has been used in many surgeries since its introduction in 2000. In 2016, a total of $753,000$ surgeries were reported worldwide, which represents a yearly increase of 15% [21] from its launch.

Haptic feedback is crucial to help surgeons feel the sensations they need to feel during surgical procedures.

Haptic feedback incorporates the sense of touch to improve soft tissue manipulation and avoid damage due to excessive applied force. Haptic sensation experienced by a skilled hand exceeds auditory and visual feedback when controlling a surgical robot. On top of that, provision of haptic feedback during surgery could help a surgeon to decide on the amount of force and torque, and the direction and orientation of the force and torque to exert on a soft tissue under consideration.

Needle insertion, suturing, knot-tying and needle passing are common procedures in robotic surgery. Needle insertion provides the mode for delivering local anesthetic drugs, to take blood sample and to perform procedures such as suturing, neurosurgery and brachytherapy [22]. In the suturing task, the surgeon picks up the needle and approaches the incision, marked as a vertical line on the bench-top model [4]. The surgeon then threads the needle through the tissue, entering at the dot on one side of the incision and exiting at the corresponding dot on the other side. After the first needle pass, the surgeon removes the needle, transfers it to the right hand, and repeats the process three more times [4]. In the knot-tying task, the surgeon takes one end of a suture attached to a flexible tube anchored at both ends to the bench-top model, and ties a single loop knot [4]. For the needle passing task, the surgeon picks up the needle and passes it through four small metal hoops positioned slightly above the surface of the bench-top model, moving from right to left [4].

Utilizing machine learning in robotic surgery realm holds promise for enhancing telehealth applications. Specifically, in the context of robotic surgery, machine learning stands to improve patient safety by ensuring accurate predictions. However, to effectively leverage machine learning models for robotic surgery application, a solid foundation of domain knowledge is imperative to enhance model performance and address the reality-model mismatch. Furthermore, machine learning applications for robotic surgery scenarios must be equipped to facilitate robust online learning, particularly for datasets with inherent uncertainty or limited size. The following publicly available data such as JIGSAW, dataset on force measurements on needle insertion into two ex-vivo human livers [23], and force measurements for needle insertions into SEBS soft tissue mimicking simulants with and without waterjet running through the needle [24] are used for training machine learning model for robotic surgery.

Three key challenges in applying machine learning to robotic surgery are expected. First, learning and prediction times must be sufficiently short to facilitate incremental model updates. This constraint may limit the applicability of machine learning models in certain scenarios, even though they offer the advantage of adapting dynamically to changing environments. Additionally, the prediction time can be adversely affected by the dimensionality of the data. The second challenge is accuracy. Applications in robotic surgery are highly sensitive to prediction errors, which can have life-threatening consequences. Striking an optimal balance between latency and accuracy is crucial, yet highly challenging. The third challenge involves the online updating of model parameters. The thesis focuses on addressing these issues in remote robotic surgery, particularly regarding latency and reliability. To tackle these challenges, we propose the use of supervised machine learning models, which inherently involve optimization tasks. Effectively overcoming these optimization challenges is vital for improving machine learning algorithms in this field. The goal of this thesis is to improve various optimization schemes for training machine learning models applied to remote robotic surgery, thereby enhancing reliability. Each aspect of these optimization schemes is critical for the success of robotic surgery. By implementing these innovative optimization strategies, patient outcomes can be improved. The advantage of utilizing optimization schemes to train machine learning models for robotic surgery is that they can converge to quality solutions even from random initializations, thereby ensuring system stability. Accuracy is essential in robotic surgery, and stability directly correlates with accuracy. Consequently, low prediction accuracy can destabilize the predictive framework, making convergence of the algorithm crucial for improving model generalization performance in robotic procedures. Moreover, it is important to create a basin of attraction that ensures the predictive framework converges to a quality solution, even from random initializations. In a nutshell, by enhancing the performance of modern optimization schemes, an improved accuracy in machine learning models for robotic surgery procedures can be achieved.

1.3 Thesis Contribution

The examination of the challenges in machine learning reveals that the existing optimization methods in modern machine learning fellshort of meeting the reliability standards necessary for remote robotic surgery. This thesis is dedicated to tackling these challenges head-on with the goal of establishing a dependable framework for remote robotic surgery procedures. It should be noted that, all the research, formulation, and experiments presented in this thesis, which amalgamate findings from the published papers, were conducted by Francis Boabang under the supervision of Dr. Roch Glitho. Francis Boabang authored the manuscript with occasional proofreading by the co-authors including Dr. Elbiaze Halima, Dr. Martin Maier, Dr. Wessam Ajib, Dr. Amin Ebrahimzadeh, Dr. Omar Alfandi, and Dr. Fatna Belqasmi, and incorporating feedback from them. Furthermore, we received abundant feedback and insightful comments from anonymous journal reviewers, which greatly contributed to shaping the published and unpublished papers utilized for this thesis.

1.3.1 An Improved Low Rank Matrix Factorization

In the first contribution, an improved low rank matrix factorization method to scale machine learning and application to remote robotic surgery is proposed [25]. In this contribution, a scalable Gaussian Process Regression to scale machine learning is advocated. Given that the standard eigen decomposition for online Gaussian Process Regression covariance update is cost-prohibitive, computationally effective matrix separation algorithms called a sequential norm regularized randomized low-rank and sparse matrix factorization (SRLSMF) and a sequential ℓ_1 norm regularized randomized low-rank and sparse matrix factorization (ℓ_1-SRLSMF) to update the covariance matrix in Gaussian Process Regression are proposed. Specifically, SRLSMF and ℓ_1-SRLSMF to scale the Gaussian Process Regression and address the issue of delayed/lost data in the training dataset in 5G-enabled Tactile Internet remote robotic surgery are utilized. The symmetric positive semidefinite constraint explicitly during the learning of SRLSMF and ℓ_1-SRLSMF to ensure the positive definiteness of the associated solutions are enforced. This allows us to find a robust low-rank approximation of the kernel matrix while maintaining its structure of positive semi-definiteness [26]. We also include a theoretical proof to show that the costs of SRLSMF and ℓ_1-SRLSMF are smaller than those of direct robust matrix factorization (DRMF) [5] and RPCA [27] in a sequential setting. Specifically, the proof that using the decomposition results of the previous iteration to compute the decomposition of the current iteration step can minimize the computational costs of DRMF [5] and robust principal component analysis (RPCA) [27] in a sequential setting is presented. By means of extensive simulations, the theoretical perspective of using both ℓ_1-SRLSMF and SLRSMF Gaussian Process Regression is justified. The simulation results show that the proposed methods can effectively recover the true low-rank matrix from the corrupted data, and thereby to predict the next action of the surgeon with minimal computational burden in Tactile Internet remote robotic surgery. The outcome of the contribution was published in IEEE transaction on Network and Service Management.

Francis Boabang, A. Ebrahimzadeh, R. Glitho, H. Elbiaze, M. Maier and F. Belqami, ”A Machine Learning Framework for Handling Delayed/Lost Packets in Tactile Internet Remote Robotic Surgery,” in IEEE Transactions on Network and Service Management, doi: 10.1109/TNSM.2021.3106577.

1.3.2 An Improved Stochastic Gradient Descent Algorithm

The partially adaptive momentum estimation method (PADAM) [3, 28, 29] is reformulated to develop a new and better ASGD method capable of overcoming the small learning rate dilemma problem by assigning different base learning rates to different categories of coordinate values. The theoretical properties of the proposed ASGD in a convex and nonconvex setting following the authors in [30], and replacing the base learning rate with a linear function to achieve an improved convergence rate $\mathcal{O}\left(\frac{1}{T^{\frac{1}{2}}} + \Psi\right)$ and $\mathcal{O}\left(\frac{d^{\frac{1}{2}}}{T^{\frac{1}{2}}} + \frac{d}{T} + \Psi\right)$ is rigorously analyzed, which represents an improvement over Amsgrad, Padam, and Adam algorithms [3,28,31,32]. Then, the theoretical properties of the proposed ASGD method through thorough evaluations conducted on CIFAR-100 and CIFAR-10 datasets utilizing WideResNet and VGG16-Net architectures are justified. The proposed ASGD is applied to surgical gesture recognition task with JIGSAW dataset to ameliorate their detection performance against state-of-the-art optimizers. The results illustrate that the proposed optimizer surpasses state-of-the-art approaches in terms of generalization performance. The outcome of this contribution " is to be submitted [33].

1.4 Thesis Outline

The remainder of the thesis is structured as follows: Chapter 2 presents the literature review. Chapter 3 introduces enhanced low-rank matrix factorization schemes designed to scale machine learning and their application to remote robotic surgery. Chapter 4 delves into an enhanced stochastic gradient descent method for training machine learning models and its application to surgical gesture recognition. Finally, the findings, the limitations of the proposed solution, and potential directions for future research are summarized.

Chapter 2

Literature Review

In machine learning, optimization algorithms like stochastic gradient descent, low-rank matrix factorization, and a range of loss functions are fundamental. Following this, an overview of these optimization schemes necessary for refining machine learning performance is discussed. This review aims to scrutinize the limitations of these models as potential optimization schemes. Finally, cutting-edge machine learning algorithms employed in robotic surgery and their limitations are examined.

2.1 Preliminaries of Low Rank Matrix Factorization

Matrices is data representation which is made up of numbers, symbols, or expressions, arranged in rows and columns used particularly in regression and classification tasks [5]. Each entry in a matrix is called an element, each row corresponds to a sample and each column represents a feature [5]. Matrices are widely used in various fields of mathematics, science, engineering, and computer science to represent and solve linear equations, perform transformations, and store data, among other applications [34]. Let's denote $G = [g_{i,j}]$ as a matrix, where $g = [g_i]$ is a column vector and g is a scalar. Additionally, let I_n be the $m \times m$ identity matrix, with diagonal entries being ones and off-diagonal entries being zeros. The column space of G refers to the set encompassing all possible linear combinations of its column vectors. Also, let $[m]$ denote the set $\{1, 2, ..., n, ..., m\}$, and $nnz(G)$ represent the number of nonzero entries of G.

The squared vector ℓ_2 norm, denoted as $||g||_2^2$, is defined as:

$$||g||_2^2 = \sum_i g_i^2 \tag{1}$$

The squared matrix Frobenius norm, denoted as $||G||_F$, is defined as:

$$||G||_F = \sum_{ij} g_{ij}^2 \tag{2}$$

And the matrix l_1 norm, denoted as $||G||_1$, is defined as:

$$||G||_1 = \max_{x \neq 0} \frac{||Gx||_1}{||x||_1} \tag{3}$$

Low-rank matrix factorization serves to acquire a low-dimensional representation of a high-dimensional kernel matrix in terms of the ℓ_2 norm. This factorization aims to minimize a cost function, measuring the disparity between matrix G and an approximation of the kernel $L_{lowrank}$, while imposing a constraint that the approximated matrix has a reduced rank [34]:

$$\begin{aligned} \min_{L} \quad & ||G - L_{lowrank}||_F, \\ \text{subject to} \quad & \text{rank}(L_{lowrank}) \leqslant r, \end{aligned} \tag{4}$$

Here, $||.||_F$ represents the Frobenius norm. Singular value decomposition (SVD) stands out as the one of the predominant approach for low-rank matrix factorization analysis. Given a matrix $G \in \Re^{m \times m}$ and a column $V \in \Re^{m \times 1}$ and $U \in \Re^{m \times 1}$, and $\Sigma \in \Re^{d}$ been the diagonal matrix, we assume SVD of G is

$$G = U \Sigma V^T. \tag{5}$$

Eigen decomposition positive definite version of SVD can be used for low-rank matrix factorization analysis. Given a matrix $G \in \Re^{m \times m}$ and a column $V \in \Re^{m \times 1}$, eigen decomposition of G is assumed.

$$G = U \Sigma U^T. \tag{6}$$

Also, QR decomposition is computed for a matrix G with dimension $m \times m$. The QR decomposition of G is computed as

$$G = Q_G \times R_G \tag{7}$$

The matrix Q_G has orthonormal columns meaning $Q_G Q_G^T = I_n$ is orthonormal. The matrix R_G is upper triangular matrix meaning all elements below the main diagonal are zero. Specifically, for all $i < j$, the (i, j)-th of R_G is zero.

The matrix factorization technique discussed above is nonconvex, which implies that without an appropriate starting point, it might struggle to reach an optimal solution. This limitation makes it unfit for employment in remote robotic surgery, where precision and reliability are crucial.

Recent advancements in the nuclear norm have led to the emergence RPCA as a method for matrix factorization [26]. RPCA stands as a significant tool for robustly recovering a low-rank matrix from corruption, such as packet loss or delay, and it facilitates the use of low-rank regularization [27]. Packet loss and delay are explicitly addressed by introducing a corruption term in the objective function and constraints. RPCA decomposes matrix G into a low-rank matrix $L_{lowrank}$ and a corruption matrix S by minimizing the ℓ_1-norm. Mathematically, the RPCA problem can be formulated as:

$$\begin{aligned} \min_{L_{lowrank}, S} \quad & ||L_{lowrank}||_* + \lambda ||S||_1, \\ \text{subject to} \quad & G = L_{lowrank} + S, \end{aligned} \tag{8}$$

Here, $||.||_*$ denotes the nuclear norm, $||.||_1$ stands for the ℓ_1-norm, and λ serves as a penalty parameter for balancing the low-rank term and the reconstruction fidelity.

2.2 Low-rank Matrix Factorization Methods in Machine Learning

Convex and nonconvex approximation methods have been studied to reduce the computational demand faced in Bayesian model while preserving its performance. Given a matrix of rank r and m data points, the sparse approximation can be represented by Gaussian Process Regression using a set of $r << m$ basis vectors, which can reduce the computational complexity to $\mathcal{O}(r^2 m)$ [16]. Low-rank matrix approximation offers an alternate solution to scale down the computational complexity of Gaussian proccess Regression [35]. A direct factorization technique was employed in [36] for a special class of kernel functions, where the covariance matrix was hierarchically factored into a product of block low-rank updates of the identity matrix, which requires $\mathcal{O}(m \log_2 m)$ operations to invert the matrix. Existing robust low-rank approximation in Gaussian Process Regression with a positive semi-definite guarantee focus on batch training the model and then using it for the autonomous navigation of a robot [27]. The kernel approximation techniques in Gaussian Process Regression have failed to account for the lost data in the training data in a sequential setting.

Both convex [37] and nonconvex [38] low-rank matrix factorization techniques have been employed to enhance the scalability of deep learning models across various tasks. For instance, Wu et al. [37] used a convex formulation of low-rank matrix factorization to scale deep learning for addressing local optima challenges in nonlinear proximal mapping between original and infrared images in small target detection. This involves optimization steps integrating proximal networks and neural layers constrained by sparsity, facilitating effective scaling. However, convex relaxed low-rank approximation, although a surrogate of the original nonconvex function, may sometimes fail to achieve the optimal solution. On the other hand, Li et al. [38] utilized the nonconvex low-rank and sparse matrix factorization method to approximate neural network weight matrices, ensuring that neural network neurons maintain their expressive capabilities. Nonetheless, nonconvex low-rank matrix factorization requires a well-chosen initial point to converge to a satisfactory solution, which can be challenging to find.

In summary, many efforts to scale machine learning models rely on nonconvex optimization techniques. Yet, these methods frequently encounter suboptimal solutions, unable to surpass the dominance of poor minima over good ones. Without good initialization conditions, achieving high-quality solutions is not possible. Consequently, such approaches are ill-suited for most machine learning applications. Moreover, even using a convex approximation of the nonconvex formulation does not yield optimal solutions, as it merely approximates the original nonconvex optimization. The proposed algorithm has a better initialization, achieving a low-rank recovery rate of $O(\frac{1}{r\gimel})$. This depict the best recovery rate among methods that used low-rank approximations to scale machine learning [25]. On the otherhand, nonconvex PCA models showed a recovery rate of $O(\frac{1}{\varrho^2 r\gimel})$ while the convex relaxed PCA (RPCA) exhibits the worst recovery rate at $O(\frac{1}{r\gimel^2})$ [1, 2, 25]. We denote ϱ as the condition number of the low rank matrix, $\gimel$ as the incoherence of the low rank matrix and r as the rank of the matrix.

2.3 Preliminaries on Stochastic Gradient Descent

Batch gradient descent involves computing the cost function concerning the parameters across the complete dataset in a single update. Moreover, this method calculates the gradient for the entire dataset simultaneously. While it ensures convergence to a local minimum for convex functions and a local minimum for nonconvex functions, it can be slow and impractical for datasets that exceed memory capacity. Given the gradient $z_t = \bigtriangledown g(W_t)$, weight matrix W and learning rate α, batch gradient descent is formulated as

$$W_{t+1} = (W_t - \alpha * z_t)\,. \tag{9}$$

Table 2.1 Summary of the related work. Our proposed algorithm boasts a superior initialization, achieving a low-rank recovery rate of $\mathcal{O}(\frac{1}{r\daleth})$.This represents the best recovery rate among methods that leverage low-rank approximations to scale machine learning. In comparison, nonconvex PCA models attain a recovery rate of $\mathcal{O}(\frac{1}{\varrho^2 r\daleth})$ whilst the convex relaxed PCA exhibits the poorest recovery rate at $\mathcal{O}(\frac{1}{r\daleth^2})$ [1, 2].Batch represents offline and incremental denotes online.

Method	**Training Time Complexity**	**Low Rank Matrix Recovery Rate**	**Address online lost data**	**Mode**	**Reference**
Sparse Gaussian Process Regression	$\mathcal{O}(r^2m)$	$\mathcal{O}(\frac{1}{r\daleth^2})$	No	Incremental	[16]
RPCA Gaussian Process Regression	$\mathcal{O}(r^2m)$	$\mathcal{O}(\frac{1}{r\daleth^2})$	No	Batch	[27]
RPCA Deep Learning	$\mathcal{O}(r^2m)$	$\mathcal{O}(\frac{1}{r\daleth^2})$	No	Batch	[37]
PCA Gaussian Process Regression	$O(r^2m)$	$\mathcal{O}(\frac{1}{\varrho^2 r\daleth})$	No	Incremental	[35]
PCA Deep Learning	$\mathcal{O}(r^2m)$	$\mathcal{O}(\frac{1}{\varrho^2 r\daleth})$	No	Batch	[38]
Proposed	$\mathcal{O}(rm)$	$\mathcal{O}(\frac{1}{r\daleth})$	Yes	Incremental	[25]

SGD stands out as one of the most widely utilized optimizers for training machine learning models. Its popularity stems from several advantages, notably its effectively utilization of a small sub-sample to update model parameters per iteration. This characteristic renders SGD computationally cost-effective, as its performance is independent of the training size, enabling scalability to large datasets and models. Consequently, SGD methods have emerged as the dominant approach for training various types of models, including deep learning, shallow learning, and non-parametric kernel models [18]. Therefore, we considered it as the optimization scheme for training machine learning models in our remote robotic surgery application.

However, despite their widespread adoption and success, SGD methods encounter challenges, particularly in terms of convergence speed [18]. While they excel in scalability and effectiveness, their convergence can be relatively slow compared to other optimization algorithms. Stochastic gradient descent uses the same learning rate for all the coordinate values as shown below

$$W_{t+1} = \left(W_t - \alpha * z_t(W, x^i, y^i)\right). \tag{10}$$

Adaptive stochastic gradient descent uses different effective learning rate for different coordinate values. The idea is to vary the step-size for different coordinate values. Additionally, tuning ASGD method parameters such as step-size can prove to be challenging, requiring careful consideration and experimentation to achieve optimal performance [18]. These complexities highlight areas where further research and optimization efforts can enhance the effectiveness of ASGD methods in machine learning applications such as remote robotic surgery. Given the adaptive parameter β, the momentum $n_t = \beta_1^t n_{t-1} + (1 - \beta_1^t) z_t$, z_t^2 represents the sum of squares of the gradient and small constant $0 < \epsilon << 1$, the adaptive gradient descent in [30] is given as

$$W_{t+1} = \left(W_t - \frac{\alpha}{\sqrt{(z_t^2 + \epsilon)}} n_t \right). \tag{11}$$

In adaptive stochastic gradient descent (SGD) [30], momentum behaves much like a ball rolling down a hill. It speeds up in dimensions where gradients align, aiding descent, while it slows down in dimensions with changing directions. This momentum parameter, denoted as β, captures the inertia-like behavior of the optimization process. The first order momentum term is computed as

$$n_t = \beta_1 n_{t-1} + (1 - \beta_1) z_t. \tag{12}$$

Adaptive stochastic gradient descent know as adagrad eliminates the need to manually tune the learning rate. However, the learning rate can become increasing small. To resolve this issue, Adadelta and RMSProp [39] introduce second order momentum term, which is computed as the preconditioner $H_t = \beta_2 z_{t-1}^2 + (1 - \beta_2) z_t^2$. The sum of squares is calculated recursively as a decaying average of all previous squared gradients. Therefore, Adadelta and RMSProp method is formulated as

$$W_{t+1} = \left(W_t - \frac{\alpha}{\sqrt{(H_t + \epsilon)}} n_t \right). \tag{13}$$

Adam [30] extends upon Adadelta and RMSProp by maintaining an exponentially decaying average of past squared gradients, denoted as H_t, similar to those methods. Additionally, it incorporates an exponentially decaying average of past gradients, denoted as H_t, akin to momentum, as illustrated below,

$$n_t = \beta_1 n_{t-1} + (1 - \beta_1) z_t. \tag{14}$$

and

$$H_t = \beta_2 H_{t-1} + (1 - \beta_2) z_t^2. \tag{15}$$

2.4 Stochastic Gradient Descent Algorithms in Machine Learning

SGD has been widely used for many applications. Still, its convergence is slow, in consequence limiting its applications. Contrary to SGD, which uses the same base learning rate for all the coordinates, adaptive SGD methods derive different effective learning rates for different coordinate values from the approximation of

Table 2.2 Summary of works related to adaptive stochastic gradient descent, where 'D_{iff1} ' denotes a base learning rate, 'D_{iff2}' denotes another base learning rate, and 's' characterize the gradient growth rate of the cumulative stochastic gradient. D_{iff1} and D_{iff2} can be selected from $\{10, 1, 0.1, 0.01, 0.001, 0.0001\}$

Optimizer	ADAM [30]	PADAM [28]	AMSGRAD [40]	Wada [41]
Large H_t	D$_{iff1}$	D$_{iff1}$	D$_{iff1}$	D$_{iff1}$
Small H_t	D$_{iff1}$	D$_{iff1}$	D$_{iff1}$	D$_{iff1}$
Rate of Convergence in Convex Setting	$\mathcal{O}\left(\frac{1}{T^{\frac{1}{2}}}\right)$	$\mathcal{O}\left(\frac{1}{T^{\frac{3}{4}-\frac{s}{2}}}\right)$	$\mathcal{O}\left(\frac{1}{T^{\frac{1}{2}}}\right)$	$\mathcal{O}\left(\frac{1}{T^{\frac{1}{2}}}\right)$
$\hat{H}_t = \max(\hat{H}_{t-1}, H_t)$	No	Yes	Yes	No
Rate of Convergence in Nonconvex Settings	$\mathcal{O}\left(\frac{d^{\frac{1}{2}}}{T^{\frac{1}{2}}} + \frac{d}{T}\right)$	$\mathcal{O}\left(\frac{d^{\frac{1}{2}}}{T^{\frac{3}{4}-\frac{s}{2}}} + \frac{d}{T}\right)$	$\mathcal{O}\left(\frac{d^{\frac{1}{2}}}{T^{\frac{1}{2}}} + \frac{d}{T}\right)$	$\mathcal{O}\left(\frac{d^{\frac{1}{2}}}{T^{\frac{1}{2}}} + \frac{d}{T}\right)$
Non-Ergodic Convergence Analysis	No	No	No	No

first and second-order momentum of the gradient [30]. Momentum's ability in accelerating convergence is primarily observed in the realm of strongly convex functions, thus it may not yield accelerated convergence rates for nonconvex objectives. Moreover, addressing the nonconvergence challenges inherent in nonconvex scenarios with adaptive stochastic gradient descent can be achieved by employing exponential moving averages of historical gradient squares. Nevertheless, the moving average approach is hindered by the short memory problem, potentially leading to failures in specific circumstances such as nonconvex setting. Addressing the short memory constraint of adaptive SGD can be achieved by integrating long-term memory mechanisms [40]. While momentum-based algorithms have attempted to address the convergence issue, the problem of a small base learning rate leading to low generalization error in the later stages of training remains unresolved.

The base learning rate selection affects the convergence rate of adaptive learning rate of ASGD. For instance, some coordinate values are small, so to avoid those coordinate values from overshooting and reducing the model's generalization performance, ADAM [30] and AMSgrad [40] selected small base learning rates [28]. Choosing a small base learning rate makes the algorithm make less impact at the later stage of training [28]. Chen et al. [28] sought to address this issue by opting for a small partially adaptive parameter alongside a large base learning rate. But, the coordinate values still overshoot since the partial adaptive value was not adapted during training, leading to poor generalization performance. Moreover, when examining different adaptive SGD methods, it is important to recognize that partially adaptive parameters below 0.5 often lead to performance outcomes devoid of any predictable pattern [28, 29]. On top of that, Sun et al. [3] introduced a method to address the challenge of small learning rates by advocating for the selection of a partially adaptive parameter greater than 0.5. Nonetheless, opting for a partially adaptive parameter exceeding one frequently results in uncertain outcome. AdaBelief algorithm in [39] scales the learning rate in adaptive function by utilizing the difference between the predicted and observed gradient. Recently, Zhou et al. [42] modified the second-order momentum of AdaBelief [39] to boost its convergence under strong convexity condition. Nevertheless, the improved AdaBelief does not exhibit the optimal rate of convergence especially for weakly convex and nonconvex objective functions. We need an approach that is independent of a particular objective function. Furthermore, an approach capable of handling both weakly convex and nonconvex objective functions can substantially enhance the convergence rate of any model. Recently, Huang et al. [43] presented an angle-calibrated moment technique that leverages the benefits of a second-order moment while updating first order moment. They compared its convergence rate with that of Adam and Padam, and the result showed a close convergence rate to Adam and Padam. Although this approach reduces the number of update parameters, it may entail a loss in model effectiveness. Chen et al. [44] implemented a mechanism which aims to enhance the empirical performance of models by gradually diminishing the cumulative impact of a gradient on all subsequent updates. Verma et al. [45] suggested employing a trigonometric function on the exponential moving average of weight parameters to calculate the step size. This approach only targets the vanishing gradient issue in nonconvex scenarios, particularly prominent when employing sigmoid activation functions. Zhong et al. [41] endeavored to address the non-convergence problem present in Adam by introducing a novel approach: a linearly growing weighted strategy that assigns varying weights to past gradients. Their method demonstrated heightened efficacy, notably when the gradient experienced rapid decreases. The limitation of Wada [41] lies in its tendency to diverge on non-convex and slowly decaying gradient problems. The authors in [46] introduced a novel adaptive gradient framework called SUPERADAM, designed to be faster and more versatile. This framework was based on a universal adaptive matrix encompassing various existing adaptive gradient forms, allowing it to integrate with momentum and variance reduction techniques seamlessly. The downside of SUPERADAM [46] lies in its variance reduction technique, necessitating a larger batch size for optimal performance.

In summary, all the above mentioned works have improve the convergence rate of ASGD to a certain extend. However, they continue to face constraints stemming from poor accuracy, resulting in high generalization error during advanced training stages. I contend that this issue primarily arises from the fact that the base learning rate in these algorithms can either become excessively small or excessively large in the later stages of training depending on the network architecture. Consequently, Adam, ASGD2 [3], Padam [28], Amsgrad [40], Wada [41] and SUPERADAM [46] are adopted as the benchmark methods. A summary of works related to ASGD in Table 2.2 and 2.3 are presented.

Table 2.3 Summary of works related to adaptive stochastic gradient descent, where 'D_{iff1}' denotes a base learning rate, 'D_{iff2}' denotes another base learning rate and 'Ψ' characterize the gradient growth rate of the cumulative stochastic gradient. D_{iff1} and D_{iff2} can be selected from $\{10, 1, 0.1, 0.01, 0.001, 0.0001\}$. In the ASGD2 algorithm, as discussed in [3], the parameter δ was set to 10^{-8}, which is very close to zero. Selecting a larger value causes divergence of the ASGD2 method. Therefore, for practical purposes, $\delta = 0$ can be assumed, ensuring both convex and nonconvex convergence rates.

Optimizer	ASGD2 [3]	Super-Adam [46]	Proposed(Improved Adam)	Proposed(Improved Amsgrad)
Large H_t	D_{iff1}	D_{iff1}	D_{iff1}	D_{iff1}
Small H_t	D_{iff1}	D_{iff1}	D_{iff2}	D_{iff2}
Rate of Convergence in Convex Setting	$\mathcal{O}\left(\frac{1}{T^{\frac{1}{2}}}\right)$	$\mathcal{O}\left(\frac{1}{T^{\frac{1}{2}}}\right)$	$\mathcal{O}\left(\frac{1}{T^{\frac{1}{2}}} + \Psi\right)$	$\mathcal{O}\left(\frac{1}{T^{\frac{1}{2}}} + \Psi\right)$
$\hat{H}_t = \max(\hat{H}_{t-1}, H_t)$	No	No	No	Yes
Rate of Convergence in Nonconvex Setting	$\mathcal{O}\left(\frac{lnT+d^2}{T^{\frac{1}{2}}}\right)$	$\mathcal{O}\left(\frac{lnT+d^2}{T^{\frac{1}{2}}}\right)$	$\mathrm{O}\left(\frac{d^{\frac{1}{2}}}{T^{\frac{1}{2}}} + \frac{d}{T} + \Psi\right)$	$\mathcal{O}\left(\frac{d^{\frac{1}{2}}}{T^{\frac{1}{2}}} + \frac{d}{T} + \Psi\right)$
Non-Ergodic Convergence Analysis	Yes	No	Yes	Yes

2.5 Review of State-of-the-Art Machine Learning Models in Robotic Surgery

In telesurgery, variants of probabilistic and non-probabilistic models have been proposed to assist the surgeon in performing a surgical task in the presence of data loss and delay. Ashirwad et al. [47] used the Gaussian mixture model/Gaussian mixture regression (GMM/GMR) to improve reliability in remote robotic surgery. Specifically, the GMM was adopted to encode a set of force/torque profiles and their corresponding parameters, followed by GMR, which was used to retrieve a generalized version of the force/torque profile for Trocar insertion[1] [47]. Delayed/lost haptic feedback information was successfully predicted via this GMM/GMR model in a local area network setting. In our recent work [20], we showed that the hidden Markov model/Gaussian mixture regression (HMM/GMR) makes better predictions compared to the GMM/GMR for remote needle insertion procedure. Specifically, the hidden Markov model was used to encode a set of force, torque, and corresponding parameters. GMR was then used to reproduce the generalized version of the force, torque, and corresponding parameters. Our work concluded that prediction of under 1 ms can be achieved with a minimal error by selecting a small number of states k. Both the GMM and the HMM have a computational complexity of $\mathcal{O}(mk)$. The K-nearest neighbor regression (KNNR) was used in [11] to train a tendon-driven serpentine surgical manipulator to perform a trans-oral surgical task. However, KNNR suffers from the problem of dimensionality for extremely high dimensional space (i.e., robots

[1]Trocar insertion is a type of laparoscopic surgery involving small incisions into the abdominal skin.

with multiple arms) without dimensionality reduction as a pre-processing step. The weighted KNNR in [11] has complexity of $\mathcal{O}(m^k)$. Osa et al. [10] employed Gaussian Process Regression with a conditional distribution of demonstrations, normalization in the time domain, and estimation concerning a reference trajectory to learn a surgical task. Gaussian Process Regression complexity increases cubically $\mathcal{O}(m^3)$ with the number of training data points m. Learning from demonstrations frameworks, such as the Gaussian Process Regression, the GMM, the KNNR, and the HMM, is dependent upon the surgeons' demonstrations to initialize the surgical task dynamics. In contrast, the reinforcement learning framework in [48] does not require initialization. reinforcement learning immediately learns and manipulates the tissue dynamics. Nieto et al. [49] developed a surgical system integrating virtual reality, enabling users to generate 3D virtual representations of patients for pre-surgery practice. The data collected from the virtual headset is then utilized by a reinforcement learning algorithm learn to autonomously execute surgical procedures, including tasks like needle picking. The drawback of using reinforcement learning in remote robotic surgery is that it requires a long timeframe for the robot to effectively perform the task. reinforcement learning has a complexity of $\mathcal{O}(m^3)$.

Neural-based methods in remote robotic surgery are discussed next. Xu et al. [11] employed an extreme learning machine to automate a trans-oral surgical task. Mayer et al. [12] used long short term memory neural network to learn the knot-tying procedure, while, Sneath et al. [50] applied the LSTM NN to simulate a semi-autonomous robotic surgery for a space exploration mission to the Moon and Mars. The computational complexity of LSTM NN and ELM is dependent upon the number of neurons h in each layer. Neural-based methods, such as ELM and LSTM NN, are known to have good accuracy, though they tend to overfit when the hidden neurons and the activation function are not selected correctly. Thanajeyan et al. [51] combined the advantages of LSTM NN and RL to develop an autonomous tensioning policy. All of the above mentioned works are offline and utilize fixed-parameter non-linear regressors, which are not suitable for online adaptation to unforeseen situations or new scenarios.

The work in [52] on remote robotic surgery focuses on online model learning and prediction. Fichera et al. [52] employed a locally weighted projection regression to train soft-tissue dynamics during laser exposure to predict the effects of laser-tissue interaction during surgery. The complexity of each local model of LWPR was $\mathcal{O}(m)$. LWPR parameters need to be tuned manually to arrive at an optimal solution, which is time-consuming. From the literature, the naive deployment of the machine learning frameworks in an online setting would incur a significant cost when there is high number of datapoints. Also, there may be delayed/lost data resulting from communication disruptions and sensor failure making the dataset incomplete. Therefore, suitable techniques to address the scalability and delayed/lost data issues in online learning are needed. Also, it failed to address the small learning rate dilemma, thus compromising the attainment of the optimal convergence rate. Additionally, all the aforementioned works relied on kinematic datasets for model training. A summary of the related work is presented in Table 2.1, which illustrates the complexities of various machine learning frameworks.

A model for gesture classification utilizes trimmed video clips, kinematic data, or a combination of both to classify gestures, aiming to enhance patient outcomes [53]. For example, Haro et al. [54] used multiple kernel learning to combine a Linear dynamic system (LDS) with a bag of features. The LDS model learns each gesture and video clip and exploits the distance between the model's parameters to classify new video clips. Also, Haro et al. used a bag of features to learn a dictionary of space-time words using the space-time features obtained from all the video clips to classify new gestures. Luongo et al. [55] also seek to enhance patient outcomes by classifying and identifying needle suture activity and gesture using a deep learning model on a video dataset.

Unlike gesture classification, which typically uses trimmed videos, surgical gesture recognition relies on the entirety of the video and/or kinematic dataset to identify specific gestures. It utilizes the start and end times of the gestures for recognition purposes [53]. Huynhnguyen et al. [53] used deep learning model to classify surgical gesture using the entire video data with modest success. Also, Selvam et al. [56] used a deep learning model to detect surgical activity in the surgical site. Both Selvam et al and Huynhnguyen et al. fallshort at improving the learning phase of surgical recognition model to boost the convergence rate of their model. Improving the training phase of machine learning for remote robotic surgery holds immense significance in improving surgical outcomes and ensuring patient safety.

In summary, the existing work on applying machine learning in remote robotic surgery is crippled by the following shortcomings: first, most of the existing works assign a fixed expert-predefined value for the machine learning model, and therefore, they do not generalize well on new data, and second, the computational complexities associated with existing machine learning methods in remote robotic surgery with parameter tuning schemes are costly. Furthermore, none of the discussed methods can used in real-time due to the high computational complexity of the algorithms as shown in Figure 2.4. An ideal machine learning model should have a low computational complexity of $\mathcal{O}(rm)$ and a recovery rate of $\mathcal{O}(\frac{1}{r\gimel})$ to meet the requirements of robotic surgery application [1, 2, 25]. Also, the convergence rate of the optimizer for training the machine learning should be $\mathcal{O}(\frac{d^{\frac{1}{2}}}{T^{\frac{1}{2}}} + \frac{d}{T} + \Psi)$.

Table 2.4 Summary of the related work. Batch represents offline and incremental denotes online.

Method	**Training Time Complexity**	**Low Rank Matrix Recovery Rate**	**Address online lost data**	**Mode**	**Reference**
GMM/GMR	$\mathcal{O}(mk)$	N/A	No	Batch	[47]
HMM/GMR	$\mathcal{O}(mk)$	N/A	No	Batch	[20]
LWPR	$O(m)$	N/A	No	Incremental	[15, 52]
ELM	$\mathcal{O}(mh_{layers})$	N/A	No	Batch	[11]
KNNR	$\mathcal{O}(m^k)$	N/A	No	Instance Based	[11]
RL	$\mathcal{O}(m^3)$	N/A	No	Batch	[48]
DeepRL	$\mathcal{O}(m^3h_{layers})$	N/A	No	Batch	[51]
LSTM	$\mathcal{O}(m^5h_{layers})$	N/A	No	Batch	[12, 50]
Gaussian Process Regression	$\mathcal{O}(m^3)$	N/A	No	Batch	[10]

Chapter 3

Improved Low-Rank Matrix Factorization for Scaling Machine Learning: A Case Study in Robotic Surgery

3.1 Introduction

With the advent of smart sensors and actuators enabled by the Internet of Things (IoT), a wide variety of machine-centric applications are becoming part of our everyday lives [57]. In 5G settings [58], the IoT aims to realize the connectivity of a large number of devices via massive machine-type communications (mMTC) with its inherent machine-to-machine (M2M) communications paradigm [59]. A natural evolutionary leap of the IoT is the so-called *Tactile Internet*, which aims to enable the remote control of real and/or virtual objects via haptic communications [60]. Haptic communication is geared towards human-to-machine (H2M) communications and envisions the transmission of haptic signals in addition to conventional audiovisual data and machine-centric traffic [61]. This development can help shift the emphasis towards humans, allowing for a more human-centric design approach [62]. Leveraging on haptic feedback, humans will play more active roles in future IoT settings and will be more involved in a group of immersive, human-in-the-loop (HITL) applications enabled by the transmission of multi-sensory traffic, including the sense of touch [63]. An interesting example of HITL-based Tactile Internet applications is remote robotic surgery, which can be viewed as a system consisting of several dependable and controllable tools that enable a skillful surgeon to execute surgery on a remotely-located patient.

Robotic surgery offers a wide range of benefits including improved precision, enhanced surgeon dexterity, reduced tremor, and improved ergonomic conditions [19]. Patients may also benefit especially since robotic surgery incurs less blood loss and pain, and therefore can result in reduced trauma and shorter recuperation time. Conventional robotic surgery occurs locally with a surgeon and a patient in close proximity. As an example, the Da Vinci Surgical system (Intuitive Surgery Inc.) has been used in many surgeries since its introduction in 2000. In 2016, a total of 753,000 surgeries were reported worldwide, which represents an annual increase of 15% [21].

Unlike conventional robotic surgery, remote robotic surgery enables a surgeon to perform surgery on a remotely located patient via the transmission of command and feedback signals through a communication network [64]. Typically, a transparent, immersive Tactile Internet telesurgery requires an ultra-high reliability of 99.999% as well as a very low latency of $\backsim 1$ ms [65]. Ideally, transparency requires that the positions and forces on the patient side domain and the surgeon side domain are equal. However, in reality, it is difficult to achieve true transparency due to communication-induced artifacts such as delay and packet loss [66]. According to [67], current deployments of 5G mobile networks can only partly meet the stringent latency and reliability requirements of the Tactile Internet, and so, communication-induced artifacts (e.g., delay, packet loss, and jitter) may have a detrimental impact on the stability and safety of a remote robotic surgery system [68]. To cope with excessive delays and/or packet loss, machine learning has recently proven to be a promising approach towards compensating for excessive delay and/or packet loss [20, 47, 65].

To supply the delayed/lost data in Tactile Internet-based remote robotic surgery, Gaussian Process Regression is a promising approach, especially given that it has shown promising results in 5G-enabled wireless traffic prediction [58] and in the modeling the inverse dynamics of robotic systems [69]. More specifically, Gaussian Process Regression can deal well with high dimensional data as it explicitly optimizes hyperparameters and encapsulates expert knowledge into the kernel function based on the Bayes theorems to maximize the marginal likelihood of getting explainable results [58, 70]. Despite their potential to enhance prediction accuracy, the computational complexity of Gaussian Process Regression is centered around the kernel matrix inversion, which increases with an increasing number of data points. Therefore, if Gaussian Process Regression are to be applicable in 5G-enabled Tactile Internet remote robotic surgery, the model update and prediction process must be able to be performed with both precision and rapidity.

Online and offline machine learning frameworks have been proposed for remote robotic surgery [10, 12]. Most of the frameworks in robotic surgery focus on offline training [12, 47], which eliminates the time required to update the model parameters online. These strategies are generally not well suited for robotic surgery where commands and sensor data are received in sequential streams and model parameters must be updated on-the-fly. We note, however, that in robotic surgery, the data generated by surgical robots are extremely diverse [4]. Offline training may cause significant inaccuracies in prediction for a multitude of reasons [71]. Notably, given that it is not possible to learn the complete state space beforehand, models that have learned offline can only approximate the model correctly in the area of the state space that is covered by the sample; the trained model would not be able to generalize beyond that region. In contrast, online learning creates a more general model for a larger space, as it is capable of adapting for time-dependent surgical robot dynamics.

The need to detect inaccuracies and changes creates the opportunity to adopt an intelligent, context-aware approach with the capability of dynamically determining an optimal strategy; thereby achieving tradeoff between reliability and latency [71]. In turn, this calls for an online model learning approach, a crucial method that allows a predictive model in changing environmental conditions to generate predictions with high accuracy. Online model learning is known to be computationally expensive [71]. Low-rank matrix factorization [35], an example of subspace learning, can be used to computationally scale online model learning [35]. Even though the low-rank matrix factorization methods can scale the machine learning frameworks computationally, they perform poorly in the presence of lost data [27]. In a fully online setting, there may be random delayed or lost data in the training dataset due to communication disruptions. Delayed/lost data from communication disruption or sensor failure can cause surgical robots to operate beyond their safe region, which could lead to the application of an excessive amount of force that may rupture vital organs.

Sparse and low-rank matrix factorization techniques and their variants, such as direct robust matrix factorization [5], randomized sparse low-rank matrix decomposition [72], and robust PCA [26], among others, have been widely explored and exploited to scale machine learning and recover corruptions in various classification and regression tasks such as image classification [73, 74], monaural speech enhancement [75], autonomous robotic control [27], and ultrasound imaging [76]. However, to the best of our knowledge, most of the recent solutions either exploit non-convex formulation [73–75] or convex formulation [76] of low-rank matrix factorization to scale machine learning. Multiple experiments have proven that non-convex formulations of low-rank matrix factorization are known to achieve a higher accuracy with a smaller computational burden than convex nuclear norm formulation (i.e., a convex envelope of the rank function) of low-rank matrix factorization machine learning models [73, 76]. However, without a good initial point, the number of the local minima of a non-convex formulation of low-rank matrix factorization machine learning models [73–75] grows so fast that it is effectively impossible to find a sufficiently good solution [5]. Such incorrect estimation of the kernel matrix can lead to an unstable situation when a surgical robot operates on a patient using an machine learning-driven non-convex formulation of a low-rank matrix factorization machine learning model.

As a remedy, we exploit a two-step technique; a nonconvex formulation of a low-rank matrix factorization with convex formulation of a low-rank matrix factorization initialization [5] to scale machine learning models. To be more specific, according to [5], we can initialize DRMF (i,e., a non-convex RPCA) by running RPCA (a convex relaxation) for a few iterations, which should make it possible to get better results than using DRMF or RPCA alone to scale a given model. Thus, by employing DRMF (with convex relaxed RPCA initialization included [76]) to scale a given machine learning model, we can avoid being stuck at the local minima and converge to a solution with smaller recovery residuals. Also, the running time of DRMF (including the initialization time) is expected to be smaller than that of applying RPCA alone under the same stopping criterion [5]. Therefore, machine learning driven DRMF models with convex relaxed RPCA initialization can deliver accurate predictions in a timely manner to ensure the stability of surgical robots and patient safetys.

The network requirements of Tactile Internet-based remote robotic surgery [65] can be incorporated into the DRMF [5, 72], used for scaling Gaussian Process Regression and for dealing with delayed/lost data in remote robotic surgery.

The naive version of DRMF [5] and RPCA [27] are suitable for batch mode operation. Specifically, DRMF and RPCA are intrinsically transductive methods [72, 77], and are thus inappropriate for latency-sensitive, remote robotic surgery applications, that require online computation (i.e., online model updates). For RPCA [27] and DRMF [5] to support sequential scenarios, we propose a sequential ℓ_1 norm regularized randomized low-rank and sparse matrix factorization and a sequential randomized low-rank and sparse matrix factorization, that use the eigen decomposition of the preceding results together with the incremental data to compute the next eigen decomposition. Essentially, SRLSMF and ℓ_1-SRLSMF machine learning models are inductive methods, having the ability to handle out-of-sample problems [72, 77, 78]. These models can be beneficial for remote robotic surgery, especially given that they allow for planning and updating dexemes [79] (e.g., velocity, position, and orientation information) contained in the gesture of a surgical task according to the change in conditions during a surgery to improve the accuracy of the model in a computationally efficient manner. Further, we constrain the kernel matrix to be symmetric positive-definite in order to guarantee the positive semi-definiteness of its solution. The symmetric positive-definite guarantee is required to ensure the stability of surgical robots. We provide a theoretical analysis of the proposed matrix separation algorithms. The proposed incremental eigendecomposition for handling delayed/or lost data is motivated by the incremental singular value decomposition (SVD) for handling uncertain data with missing values [80].

The novelty of this work is to exploit sparse and low-rank matrix factorization in Gaussian Process Regression to build an online model learning method to support latency-sensitive remote robotic surgery applications. While individual components of our proposed framework may have been studied extensively in the context of machine learning, their combination gives rise a computationally efficient, robust online learning method, which has the potential of being applicable in real-world Tactile Internet deployments.

The rest of the chapter are discussed as follows: First, we introduce the system model and outline the problem definition. Next, we present a scalable and robust Gaussian Process Regression framework. Following this, we detail our simulations and the results obtained. Finally, we provide conclusions.

3.2 System Model, Motivating Scenario and Problem Definition

In this section, we start by presenting the system model, followed by a motivating scenario, and then, we present our problem definition. For completeness, a list of notations is provided in Table 2.

3.2.1 System Model

Figure 3.1 illustrates the generic architecture of the bilateral remote robotic surgery system enabled by a 5G Tactile Internet wide area network (WAN). A typical remote robotic surgery system contains the following three domains: (i) the surgeon-side domain, (ii) the patient-side domain, and (iii) the network domain. The surgeon side domain consists of the surgeon along with the surgeon console, and the patient-side domain consists of the patient and a patient-side robot. The surgeon manipulates the surgeon console to generate haptic commands, which are then taken by the patient-side robot to perform the corresponding action on the patient [81]. The patient side robot interprets and uses the commands from the surgeon console in the form of position, orientation, and velocity to perform operations on the patient [81]. The network domain is responsible for offering a low latency and ultra-reliable connectivity between the patient and surgeon domains [64]. The edge/edge connector interconnects the domains via the 5G-enabled Tactile Internet WAN. We deploy a machine learning model hosted in the patient side domain for haptic command prediction.

3.2.2 Motivating Scenario

Let us focus on the knot-tying task, which is typically carried out during tissue reconstruction [4]. The knot-tying task is segmented into various gestures. Knot-tying is achieved by executing a sequence of the following

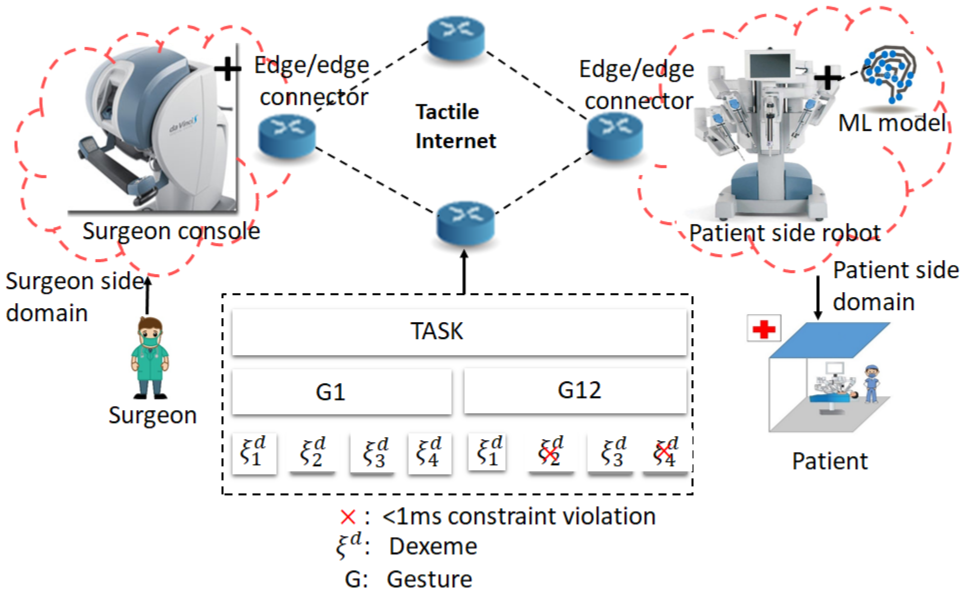


Figure 3.1 Generic architecture of our 5G-enabled Tactile Internet remote robotic surgery system.

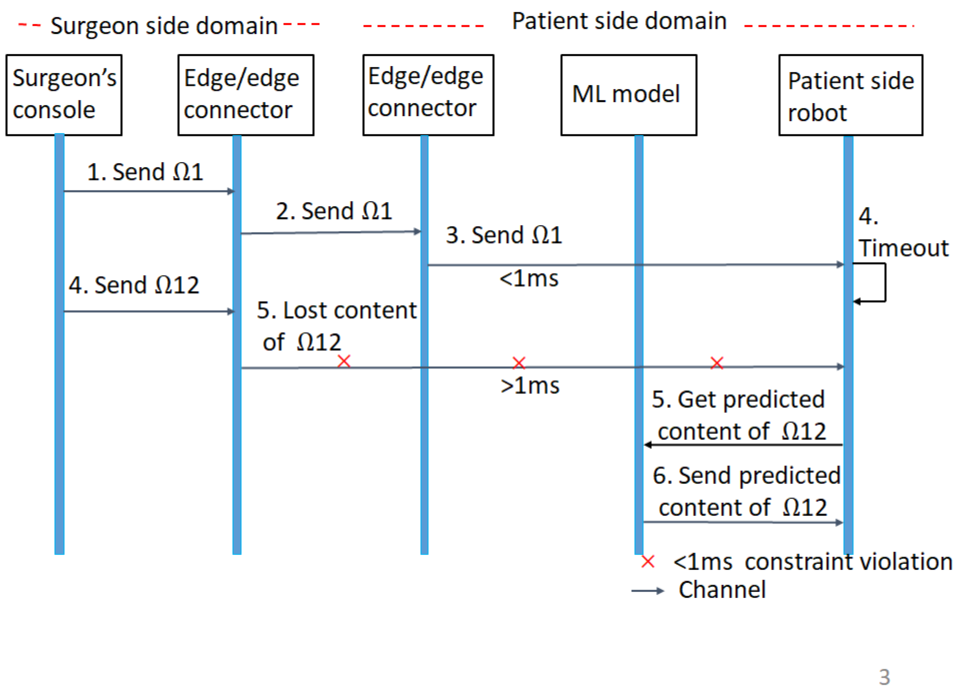


Figure 3.2 Motivating scenario.

Table 3.1 Knot tying gesture vocabulary [4].

Gesture Index	Gesture Description
$\Omega 1$	The surgeon picks up needle with right hand
$\Omega 12$	The surgeon picks up needle with left hand
$\Omega 13$	The surgeon makes a C loop around right hand
$\Omega 14$	The surgeon picks up suture with right hand
$\Omega 15$	The surgeon pulls suture with both hands to tie a knot

five sub-tasks [4]: a surgeon ($\Omega 1$) picks up a needle with her right hand, ($\Omega 12$) picks up the needle with her left hand, ($\Omega 13$) makes a so-called C loop around her right hand, ($\Omega 14$) picks up the suture with the needle in her right hand, and ($\Omega 15$) pulls the suture with both hands to tie a knot. The process is repeated until the knot-tying task is completed. Table 3.1 provides a brief description of these gestures. Surgemes are further segmented into dexemes, which are numerical versions of the sub-gestures needed to execute a surgeme [79]. Dexemes are kinematic data, such as the tool tip position, rotation matrix, gripper angle velocity, rotational velocity [4], among others. These dexemes arrive sequentially and newly arrived dexemes append to old dexemes of a surgeme. In the scenario diagram shown in Fig. 3.2, the complete dexemes contained in $\Omega 1$ are successfully sent via the 5G-enabled Tactile Internet to the patient side robot. Based on the dexemes contained in $\Omega 1$, a sequence of actions is performed by the patient-side robot on the patient. Upon receiving $\Omega 1$, the patient-side robot sets a waiting threshold, which is the period within which the complete dexemes of the next surgeme should be received. The dexemes contained in $\Omega 12$ are sent via the 5G-enabled Tactile Internet to the patient-side robot. When a content of $\Omega 12$ fails to reach the patient side robot within the pre-set threshold, the patient-side robot asks the machine learning model to send the predicted content of $\Omega 12$. Any failure of $\Omega 12$ implies the loss of certain content of $\Omega 12$, as depicted in Fig. 3.2. The dexemes in the predicted $\Omega 12$ are then used to execute the surgeon's requested action on the patient. The closeness to a real-time haptic command is dependent upon the prediction accuracy of the content of $\Omega 12$, considering that the maximum permissible deadline is 1 ms [81].

3.2.3 Problem Definition

A surgeon in the surgeon domain manipulates a patient-side robot to generate multiple demonstrations of the knot-tying task. Each knot-tying session d_{knot} is composed of R-dimensional spatial information (i.e., position, orientation, velocity, etc.) and 1-dimensional temporal information (i.e., time), where $d_{knot} \in \{1, ..., D_{knot}\}$ and D_{knot} represents the total number of demonstrations. We are given D_{knot} examples of a surgeon performing a knot-tying task with an input $\phi_m^{(d_{knot})}$ per step m and output $\xi_m^{(d_{knot})}$ per step m of length $M^{(d_{knot})}$, where $m \in \{1, ..., M^{(d)}\}$. We denote n as the index of another step of the knot tying session. Each knot-tying session is a vector ψ, composed of temporal $\phi_m^{(d_{knot})}$ and spatial information $\xi_m^{(d_{knot})}$ per step. The dataset is represented as $\psi = \{\phi_m^{(d_{knot})}, \xi_m^{(d_{knot})}\}_{m=1}^{M^{(d_{knot})}}$. The temporal information vector $\phi^{(d)} \in \Re$ serves as the input to the regression function and the spatial information vector $\xi^{(d_{knot})} \in \Re^R$ serves as the output of the knot-tying task. The Cartesian position $\overrightarrow{p} = (p_x, p_y, p_z)$, Cartesian linear velocity $\overrightarrow{v} = (v_x, v_y, v_z)$, Cartesian angular velocity $\overrightarrow{v'} = (\varphi', \pi', \varepsilon')$, 9-variable rotational matrix Z, and a gripper angle θ are the samples outputs $\xi^{(d_{knot})}$ being sent in the command path to the patient side robot and which are subject to delay and packet loss. This set of samples is used to train the machine learning model.

3.3 A Scalable and Robust Gaussian Process Regression Framework

In this work, we employ robust low-rank matrix factorization methods to achieve computationally efficient Gaussian Process Regression to facilitate haptic command prediction in a 5G-enabled Tactile Internet remote knot tying procedure. After describing the background on Gaussian Process Regression model compression via low rank and sparse matrix factorization, we present two variants of sequential randomized low-rank and sparse matrix factorization methods.

3.3.1 Background on Gaussian Process Regression Model Compression

A Gaussian Process Regression is a distribution over functions characterized by a mean $\mu(\phi^{(d_{knot})})$ and covariance $T(\phi^{(d_{knot})})$ function [10, 82]. We consider Gaussian Process Regression function $f(\phi^{(d_{knot})})$ given by

$$\begin{aligned}
&f(\phi^{(d_{knot})}) \sim \mathcal{GP}\left(\mu(\phi^{(d_{knot})}), G(\phi^{(d)}, \phi^{(d')})\right), \\
&\mu(\phi^{(d_{knot})}) = E\left[f(\phi^{(d_{knot})})\right], \\
&G(\phi^{(d_{knot})}, \phi^{(d'_{knot})}) = \\
&E\left[\left(f(\phi^{(d_{knot})}) - \mu(\phi^{(d_{knot})})\right)\left(f(\phi^{(d'_{knot})}) - \mu(\phi^{(d'_{knot})})\right)\right],
\end{aligned} \tag{16}$$

where G is the kernel function. The mean is centered at the origin because any change in actuation requires a zero mean. In Gaussian Process Regression, the target $\xi^{(d)}$ can be modelled by a Gaussian distribution as follows:

$$\begin{aligned} \xi^{(d_{knot})} &= f(\phi^{(d_{knot})}) + \epsilon, \\ \xi^{(d_{knot})} &\sim \mathcal{N}(0, G(\phi^{(d_{knot})}, \phi^{(d_{knot})}) + \sigma_n^2 I), \end{aligned} \tag{17}$$

where ϵ is the Gaussian noise, which is an independent and identically distributed random variable with zero mean and variance $\sigma_n^2 I$ [70]. A popular kernel choice is the square exponential (SE) kernel also known as the Radical basis function. Typically, an SE kernel is chosen in most Gaussian Process Regression applications, as it requires computing relatively few parameters. The formula for SE is given by

$$G\left(\phi^{(d_{knot})}, \phi^{(d'_{knot})}\right) = \sigma_s^2 \text{exp}\left(\frac{1}{2}\left(\phi^{(d_{knot})} - \phi^{(d'_{knot})}\right)^T \mathcal{W}\left(\phi^{(d_{knot})} - \phi^{(d'_{knot})}\right)\right), \tag{18}$$

where σ_s^2 denotes the variance of the kernel function itself and W is a diagonal matrix. The joint distribution of the observed target values $\xi^{(d_{knot})}$ and predicted values $\xi^{(d_{knot}*)}$ for a query point $\phi^{(d_{knot}*)}$ can be modeled as a multivariate Gaussian as follows:

$$\begin{aligned} &\wp \begin{pmatrix} \xi^{(d_{knot})} \\ \xi^{(d_{knot}*)} \end{pmatrix} \sim \\ &N\left(0, \begin{bmatrix} G\left(\phi^{(d_{knot})}, \phi^{(d_{knot})}\right) + \sigma_n^2 I & G\left(\phi^{(d_{knot})}, \phi^{(d_{knot}*)}\right) \\ G\left(\phi^{(d_{knot}*)}, \phi^{(d_{knot})}\right) & G\left(\phi^{(d_{knot}*)}, \phi^{(d_{knot}*)}\right) + \sigma_n^2 I \end{bmatrix}\right), \end{aligned} \tag{19}$$

where $\phi^{(d_{knot})}$ and $\phi^{(d_{knot}*)}$ represent all the input points and predicted values of the query points, $G(\phi^{(d_{knot})}, \phi^{(d_{knot})})$ denotes the covariance function, also known as the kernel function, and $G(\phi^{(d_{knot})}, \phi^{(d_{knot}*)})$ represents the covariance matrix between the test points and full training points. The parameters estimated via the maximum likelihood approach are the parameters of the kernel function: the variance σ_n^2 and the diagonal components of the matrix $\mathcal{W}$, such that the marginal log-likelihood is maximized [7]. The conditional distribution of the Gaussian demonstration (19) produces the predicted mean μ^* and covariance T^*_{cov} as follows:

$$\mu^* = G\left(\phi^{(d_{knot}*)}, \phi^{(d_{knot})}\right)\left(G\left(\phi^{(d_{knot})}, \phi^{(d)_{knot}}\right) + \sigma_n^2 I\right)^{-1} \xi^{(d_{knot})}, \tag{20}$$

$$\begin{aligned} T^*_{cov} = G\left(\phi^{(d_{knot}*)}, \phi^{(d_{knot}*)}\right) + \sigma_n^2 I - G\left(\phi^{(d_{knot}*)}, \phi^{(d_{knot})}\right) \\ \left(G\left(\phi^{(d_{knot})}, \phi^{(d_{knot})}\right) + \sigma_n^2 I\right)^{-1} G\left(\phi^{(d_{knot})}, \phi^{(d_{knot}*)}\right), \end{aligned} \tag{21}$$

$$\vartheta^{-1} = \sigma_n^{-2} I - \sigma_n^{-4} U \left(\Sigma^{-1} + \sigma_n^{-2} I \right)^{-1} U^T \tag{22}$$

where $\vartheta = G\left(\phi^{(d)}, \phi^{(d)}\right) + \sigma_n^2 I$ is the regularized covariance matrix. Typically ϑ is dense, causing the standard direct inversion and determinant evaluation to require $\mathcal{O}(m^3)$ operations [82]. Gaussian Process Regression is known to achieve higher performance with minimal parameter tuning. However, it suffers from significantly higher complexity. Next, we propose to explore two variants of robust kernel low-rank matrix factorization methods to address the scalability and delayed/lost data issues of Gaussian Process Regression in sequential setting under varying latency.

Kernel low-rank matrix factorization is used to obtain a low-dimensional representation of a high dimensional kernel matrix in an ℓ_2 norm sense. Kernel low rank matrix factorization can be interpreted as minimizing a cost function, measuring the fit between the kernel matrix G and an approximation of the kernel $L_{lowrank}$ subject to the constraint that the approximated matrix has a reduced rank:

$$\begin{aligned} \min_{L_{lowrank}} \quad & ||G - L_{lowrank}||_F, \\ \text{subject to} \quad & \text{rank}(L_{lowrank}) \leqslant r, \end{aligned} \tag{23}$$

where $||.||_F$ is a Frobenius norm. Based on the recent advances made in the nuclear norm, a method for kernel matrix factorization known as robust kernel principal component analysis has emerged [26]. RKPCA is one of the most significant tools for robustly recovering a low-rank matrix from corruption (packet loss/ delay), and it opens the door for using low rank regularization [27]. Packet loss and delay are considered explicitly introducing a corruption term in the objective function and in the constraints. RKPCA decomposes the kernel matrix G into low-rank matrix $L_{lowrank}$ and corruption matrix S by minimizing the ℓ_1-norm. Mathematically, the problem of RKPCA can be solved by using the following formulation:

$$\begin{aligned} \min_{L_{lowrank}, S} \quad & ||L_{lowrank}||_* + \lambda ||S||_1, \\ \text{subject to} \quad & G = L_{lowrank} + S, \end{aligned} \tag{24}$$

where $||.||_*$ is the nuclear norm, $||.||_1$ is the ℓ_1-norm, and α is a penalty parameter for balancing the low-rank term and the reconstruction fidelity. For some orthogonal matrix $U \in \Re^{m \times r}$ and diagonal matrix $\Sigma \in \mathcal{D}^r$ with entries ϱ_i, problem (24) is transformed into problem (33) to ensure the symmetric positive semi-definiteness of the kernel matrix [26, 27].

$$\begin{aligned} \min_{U,\Sigma,\hat{\Sigma},P} \quad & ||G - U\hat{\Sigma}U^T||_1 + \lambda||\Sigma||_*, \\ \text{subject to } \Sigma \geq 0, \quad & UU^T = I_r, \\ & \hat{\Sigma} = \Sigma, \quad P = U\hat{\Sigma}U^T, \end{aligned} \tag{25}$$

where $\hat{\Sigma}$ and P are two auxiliary variables. More details about the framework are presented below. Using augmented Lagrangian framework [83], we transformed the constrained problem (33) into an unconstrained Lagrangian problem below:

$$\begin{aligned} \mathcal{L}(G, P, U, \Sigma, \hat{\Sigma}) = ||G - P||_1 + \lambda||\Sigma||_* + \text{tr}\left(\Lambda_1^T\left(P - U\hat{\Sigma}U^T\right)\right) + \\ \text{tr}\left(\Lambda_2^T\left(\hat{\Sigma} - \Sigma\right)\right) + \frac{\beta}{2}\left(||P - U\hat{\Sigma}U^T||_F^2 + ||\hat{\Sigma} - \Sigma||_F^2\right), \end{aligned} \tag{26}$$

where $\Lambda_1, \Lambda_2 \in R^{m\times m}$ are Lagrange multipliers and $\beta > 0$ is a weight parameter. We solve for Σ by fixing the other variables and solving the optimization problem below

$$\Sigma^* = \arg\min_{\Sigma} \frac{\lambda}{\beta}||\Sigma||_* + \frac{1}{2}||\hat{\Sigma} - \Sigma + \frac{\Lambda_2}{\beta}||_F^2, \tag{27}$$

whose solution takes the form

$$\Sigma^* = Qdiag\left[\max\left(\zeta - \frac{\lambda}{\beta}, 0\right)\right]Q^T, \tag{28}$$

where Q is some orthogonal matrix and $max(.,.)$ should be understood as element-wise. Given $H_{gpr} = \hat{\Sigma} - \frac{\Lambda_2}{\beta}$, then the eigendecompositon of H_{gpr} is $H_{gpr} = Q\Gamma Q^T$ for $\Gamma = diag(\zeta)$.

Next, we update U_{eigen} by fixing the other variable and solving the problem below

$$U^* = \arg\min_{U} \text{tr}\left(\left(\Lambda_1^T\left(P - U\hat{\Sigma}U^T\right)\right)\right) + \frac{\beta}{2}||P - U\hat{\Sigma}U^T||_F^2, \tag{29}$$

and then, we update $\hat{\Sigma}$ by solving the problem below

$$\begin{aligned} \hat{\Sigma}^* = \arg\min_{\hat{\Sigma}} \text{tr}\left(\Lambda_1^T\left(P - U\hat{\Sigma}U^T\right)\right) + \\ \text{tr}\left(\Lambda_2^T\left(\hat{\Sigma} - \Sigma\right)\right) + \frac{\beta}{2}\left(||P - U\hat{\Sigma}U^T||_F^2 + ||\hat{\Sigma} - \Sigma||_F^2\right). \end{aligned} \tag{30}$$

Next, we update P by solving the optimization problem below

$$\begin{aligned} P^* &= \arg\min_{P} ||G - P||_1 + \operatorname{tr}\left(\Lambda_1^T \left(P - U\hat{\Sigma}U^T\right)\right) + \\ &\frac{\beta}{2}||P - U\hat{\Sigma}U^T||_F^2. \end{aligned} \tag{31}$$

Finally, we update Lagrange multipliers Λ_1, Λ_2 as follows

$$\begin{aligned} &\Lambda_1 \leftarrow \Lambda_1 + \beta\left(P - U\hat{\Sigma}U^T_{eigen}\right), \\ &\Lambda_2 \leftarrow \Lambda_2 + \beta\left(\hat{\Sigma} - \Sigma\right). \end{aligned} \tag{32}$$

The process is repeated until convergence occurs and the corruption is minimized.

The scheme in [27] relies on some relaxation approaches to relax the rank constraint to the trace norm and the ℓ_0-norm constraint to the ℓ_1-norm, which may have a significant effect on the prediction accuracy. In contrast, the DRMF scheme proposed in [5] minimizes the error of the low-rank matrix approximation by considering the fact that delayed/lost data are small, and does so without using any relaxation approaches. By assuming a small amount of delayed/lost data in the observation matrix G, the authors of [5, 84], and [72] disregarded some data, classifying it as delayed/lost data by solving the following direct robust matrix factorization problem:

$$\begin{aligned} \min_{L_{lowrank}, S} \quad & ||(G - S) - L_{lowrank}||_F, \\ \text{subject to} \quad & \operatorname{rank}(L_{lowrank}) \leqslant r, \\ & ||S||_0 \leqslant \gamma, \end{aligned} \tag{33}$$

where γ represents the limit of the matrix of packet loss/delay. The above assumption that we can ignore some data as delayed/lost data by solving Eq. (33) can be used to incorporate packet loss/delay knowledge from the 5G-enabled Tactile Internet [85] into the model. The optimization problem (33) can be solved alternatively by solving the following the matrix factorization problem and the delayed/ lost data detection problem, until convergence occurs [72]:

$$\begin{cases} L = \arg\min_{rank(L) \leqslant r} ||X_{clean} - L||_F, \quad X_{clean} = G - S, \\ L = U\Sigma U^T, UU^T = I_r \\ S = \arg\min_{card(S) \leqslant \gamma} ||E - S||_F, \quad E = G - L \end{cases} \tag{34}$$

where I_r is an $r \times r$ identity matrix. The DRMF problem above consists of a kernel low-rank approximation and a packet loss/delayed data detection problem, both needs to be solved until convergence. The packet loss/delay data detection problem detects the delayed/lost packets per iteration step until convergence occurs. When convergence occurs, the abnormally large data values are detected as lost packets and are discarded from the kernel matrix $X_{clean} = G - S$. The number of packets that can be discarded is constrained by the packet loss/delay requirement γ. To solve the kernel low-rank approximation problem, the cleared kernel matrix X_{clean} is subject to eigen decomposition to repeatedly separate the kernel matrix into an orthogonal matrix U and a diagonal matrix Σ. The eigen decomposition of the cleared kernel matrix X_{clean} is used to update the covariance matrix of the Gaussian Process Regression.

From a computational perspective, there are various cutting edge algorithms for minimizing the DRMF problem (34) that outperform popular off-the-shelf algorithms. The work in [5] employ a block coordinate strategy [86] to iteratively solve the problem (34). With a priori knowledge about the rank r, the steps of the algorithm are described below.

To update Σ, we fix the other variables and solve the following problem:

$$\Sigma^* = \arg\min_{\Sigma} ||X_{clean} - U\Sigma U^T||_F, \tag{35}$$

and then, we update U by fixing Σ and solving the following problem:

$$U^* = \arg\min_{U} ||X_{clean} - U\Sigma U^T||_F. \tag{36}$$

Next, we solve for E as follows:

$$E = G - U\Sigma U^T, \tag{37}$$

and update S by solving the packet loss/delay problem:

$$S^* = \arg\min_{card(S) \leqslant \gamma} ||E - S||_F. \tag{38}$$

Finally, the eigen decomposition of X can be computed.

$$X = U\Sigma U^T. \tag{39}$$

The process is repeated until convergence occurs.

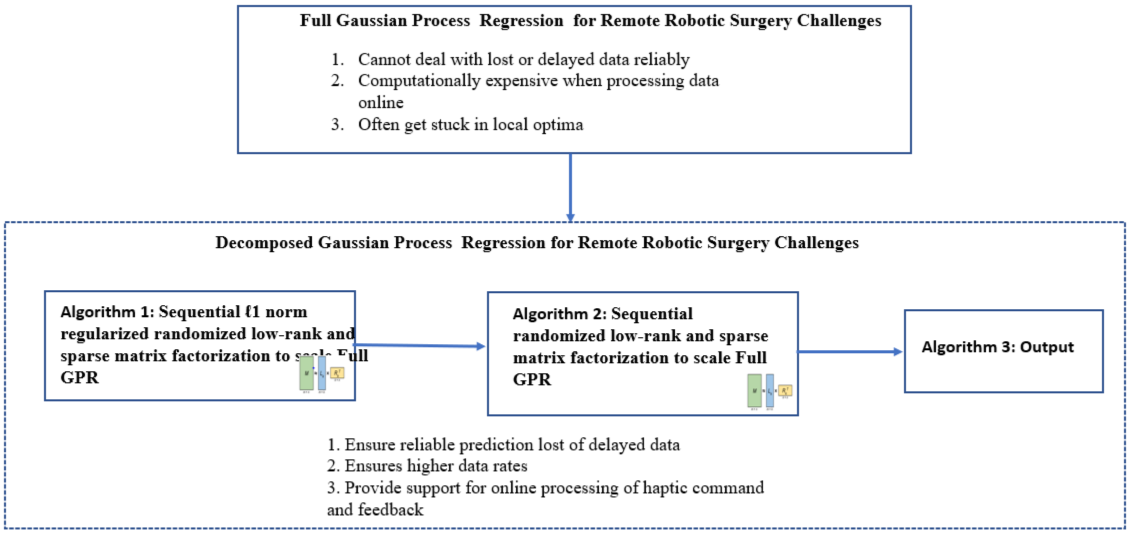

Full Gaussian Process Regression for Remote Robotic Surgery Challenges
1. Cannot deal with lost or delayed data reliably
2. Computationally expensive when processing data online
3. Often get stuck in local optima
Decomposed Gaussian Process Regression for Remote Robotic Surgery Challenges
Algorithm 1: Sequential ℓ1 norm regularized randomized low-rank and sparse matrix factorization to scale Full GPR
Algorithm 2: Sequential randomized low-rank and sparse matrix factorization to scale Full GPR
Algorithm 3: Output
1. Ensure reliable prediction lost of delayed data
2. Ensures higher data rates
3. Provide support for online processing of haptic command and feedback

Figure 3.3 Flowchart of the proposed algorithm

Remark 1. *The performance of Algorithm 2 is guaranteed when the packet losses and delays are not too high. In case of a very high packet loss and delay scenario, the imputation of the missing entries of the training dataset while performing the matrix factorization must be added.*

Remark 2. *The figure 3.3 illustrates the flowchart of the proposed algorithm, detailing its sequence of operations. The algorithm begins with a convex formulation of low-rank matrix factorization as the initialization scheme to prevent the model from getting stuck in a local optima. This is followed by a nonconvex formulation of the low-rank matrix factorization, which serves as the main optimization technique for updating the covariance matrix of Gaussian Process Regression in tactile internet applications. The goal of this proposed algorithm is to meet the stringent requirements of tactile internet, specifically for remote robotic surgery.*

3.3.2 Robust Incremental Eigen Decomposition

Online model learning for remote robotic surgery requires models to continuously adapt to new training examples over time. Incremental eigen decomposition provides a means to incorporate new data into old data without computing the matrix from scratch at every iteration. This effort and time-savings allows the dexemes of a gesture in remote robotic surgery under dynamic conditions to be planned in real time. It should be noted that we exploit incremental versions of RPCA and DRMF (i.e., SRLSMF [72] and ℓ_1 SRLSMF [77]) to scale Gaussian Process Regression. SRLSMF and ℓ_1 SRLSMF are inductive, meaning that they can cooperate with a subject with incremental eigen decomposition and work with movable and flexible objects such as

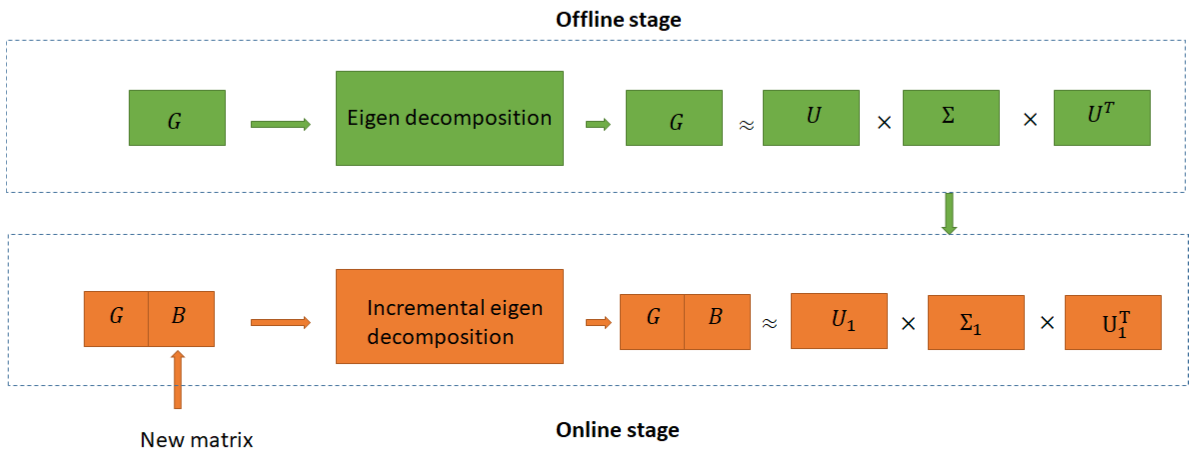


Figure 3.4 Flowchart of the offline and online stages of updating eigen decomposition. In the offline stage, a kernel matrix G is approximated by an orthogonal matrix U and a diagonal matrix Σ. In the online stage, a new matrix B is appended to the original kernel matrix G which is approximated by using the previous decomposition result of G together with the newly incorporated matrix B to approximate the resultant matrix by means of an orthogonal matrix U_1 and a diagonal matrix Σ_1.

threads and soft tissues during operations. For instance, the surgical knot-tying task involves creating loops around a surgical instrument with a surgical thread [10]. When a surgeon manipulates a surgical instrument to be looped and it is moving the dexemes of a gesture (i.e., the model parameters) must be recomputed in a computationally efficient manner to adapt the learned model according to the motion of the instrument. The inductive behavior of the proposed models ensures that the topological features of the gesture are maintained. Summaries of the exploited sequential ℓ_1 norm regularized low-rank and sparse matrix factorization and low-rank and sparse matrix factorization are presented in Algorithm 1 and 2, respectively.

Given a matrix $G \in \Re^{m \times m}$ and a column $B \in \Re^{m \times 1}$, we assume eigen decomposition of G is

$$G = U \Sigma U^T. \tag{40}$$

After appending the newly arrived data B, the larger outer matrix G_1 is built by appending columns to U. The updated matrix G_1 becomes

$$G_1 = [G|B]. \tag{41}$$

The eigenvalue and eigenvector of $G_1 G_1^T$ is required to compute the eigen decomposition of G_1. Let us denote the column matrix by $B = UF$, where F is a representation of B with U basis matrix. Given the orthogonal matrix U, the matrix F can be computed as $F = U^T B$. With $\Sigma^2 + FF^T$ being a positive symmetric matrix, we can derive the following

$$\begin{aligned} G_1G_1^T &= [G|UF][G|UF]^T, \\ &= GG^T + UF(UF)^T, \\ &= U(\Sigma^2 + FF^T)U^T, \\ &= UQ\Sigma_1^2Q^TU^T, \\ &= U_1\Sigma_1^2U_1^T. \end{aligned} \tag{42}$$

where U_1 is a rotation of U by Q.

By applying the well-known spectral theorem [87], matrix $\Sigma^2 + FF^T$ can be diagonalized using a unitary U. If the arrived data does not span the column space U then $G_1G_1^T$ becomes

$$\begin{aligned} G_1G_1^T &= [G|UF + B_1][G|UF + B_1]^T, \\ &= GG^T + (UF + B_1)(UF + B_1)^T, \\ &= [U|Q_1]\begin{bmatrix} \Sigma^2 + FF^T & FR_1^T \\ R_1F^T & R_1R_1^T \end{bmatrix}\begin{bmatrix} U^T \\ Q^T \end{bmatrix}, \\ &= [U|Q_1]Q_2\Sigma_1^2Q_2^T\begin{bmatrix} U^T \\ Q^T \end{bmatrix}, \\ &= U_1\Sigma_1^2U_1^T, \end{aligned} \tag{43}$$

where $B = UF + B_1$ and the orthogonal matrix $B_1 = Q_1R_1$ is the QR decomposition of B_1. The cost of QR decomposition of B_1 is $\mathcal{O}(ml^2)$, where l is the rank of a subset matrix. The cost of rotating U to U_1 is $\mathcal{O}(m(r+l)^2)$, the cost of obtaining F and B is $\mathcal{O}(mlr)$ and the cost to obtain Σ_1^2 is $\mathcal{O}((r+l)^3)$. The overall run time to compute SVD(G_1) is $(\mathcal{O}(ml^2) + \mathcal{O}(m(r+l)) + \mathcal{O}((r+l)^3) + \mathcal{O}(mlr))$, which is smaller than SVD in DRMF [5] and naive $RKPCA$ in [26] which has a complexity of $\mathcal{O}(\min\{mn^2, nm^2\})$.

As the matrix G grows, it becomes computationally expensive to compute the eigen decomposition of G_1. The theorem below summarizes the approximate solution to minimize the computational cost.

Theorem 1. *Let $G = U\Sigma U^T$ with $G_1 = [G|B]$ where B has no components in i-th column of U for $i > r$. The eigen decomposition of G_1 has the same spectrum ϱ_i and eigen vector u_i for $i > r$.*

Given U_1 as the first r column of U with B having U with no component U_2 and $B = UF$ for some F.

Algorithm 1 Sequential ℓ_1 norm regularized randomized low-rank and sparse matrix factorization

```
1: Input: G: the kernel matrix, rank $r$, $\lambda = 10^{-3}$, $\beta = 10^{-5}$, and $\rho = 2$
2: $j$: iterative step
3: $j = 1, U_{eigen,1} = \Sigma_1 = \hat{\Sigma}_1 = P_1 = 0, \beta_{max} = 10^{10}$,   eigen decomposition$(G_j) = U_{eigen,1}\Sigma_1 U_{eigen,1}^T$
4: Output: $U$, $\Sigma$, $\hat{\Sigma}$, $P$
5: WHILE not converged
6:     WHILE not converged
           $\begin{cases} \text{Update} \quad \Sigma_{j+1} \quad \text{by} \quad (27), \\ \text{Update} \quad U_{eigen,j+1} \quad \text{by} \quad (29), \\ \text{Update} \quad \hat{\Sigma}_{j+1} \quad \text{by} \quad (30), \\ \text{Update} \quad P_{j+1} \quad \text{by} \quad (31), \\ \text{Update eigen decomposition of} \\ G_{j+1} \quad \text{by applying Theorem 1.} \end{cases}$
7:     ENDWHILE
       $\begin{cases} \text{Update the Lagrange multiplier} \\ \quad \Lambda_1 \quad \text{and} \quad \Lambda_2 \quad \text{by} \quad (32), \\ \text{Update} \quad \beta = \min(\rho\beta, \beta_{max}) \\ j = j + 1; \end{cases}$
8: ENDWHILE
```

Algorithm 2 Sequential randomized low-rank and sparse matrix factorization

```
1: Input: G: the kernel matrix
2: $\gamma$: the maximal acceptable delay/ packet loss
3: $S$: matrix of delay/packet loss
4: $j$: iterative step
5: $j = 1$, $S_j = S$, $X_j = G_j - S$, eigen decomposition   $(X_{clean,j}) = U_1\Sigma_1 U_1^T$
6: Output: $U_{eigen}$, $\Sigma$, $S$
7: WHILE not converged
       $\begin{cases} \Sigma_{j+1} = \arg\min_{\Sigma} ||X_{clean,j} - U_{eigen,j}\Sigma_j U_j^T||_F, \\ U_{j+1} = \arg\min_{U} ||X_{clean,j} - U_j\Sigma_j U_j^T||_F, \\ E_j = G_j - U_{j+1}\Sigma_{j+1}U_{j+1}^T, \\ S_{j+1} = \arg\min_{card(S)\leqslant\gamma} ||E_j - S||_F, \\ \text{Update eigen decomposition of} \\ \text{X}_{clean,j+1} \text{ by applying Theorem 1.} \end{cases}$
8: $j = j + 1$;
9: ENDWHILE
```

Proof. From the definition of eigen decomposition in Eq. (40) we have

$$G = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix} \begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix}. \tag{44}$$

We can derive $G_1G_1^T$ as follows:

$$\begin{aligned}
G_1G_1^T &= [G|B][G|B]^T, \\
&= GG^T + BB^T, \\
&= \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma_1^2 + FF^T & 0 \\ 0 & \Sigma_2^2 \end{pmatrix} \begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix}, \\
&= \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} Q\bar{\Sigma}_1 Q^T & 0 \\ 0 & \Sigma_2^2 \end{pmatrix} \begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix}, \\
&= \begin{pmatrix} U_1Q & U_2 \end{pmatrix} \begin{pmatrix} \bar{\Sigma}_1^2 & 0 \\ 0 & \Sigma_2^2 \end{pmatrix} \begin{pmatrix} Q^TU_1^T \\ U_2^T \end{pmatrix}, \\
&= \begin{pmatrix} \bar{U}_1 & U_2 \end{pmatrix} \begin{pmatrix} \bar{\Sigma}_1^2 & 0 \\ 0 & \Sigma_2^2 \end{pmatrix} \begin{pmatrix} \bar{U}_1^T \\ U_2^T \end{pmatrix},
\end{aligned} \tag{45}$$

where U is considered unitary. From Eq. (45), Σ_2 and U_2 remain unchanged provided that $U_2^T B = 0$. For $i > r$, the spectrum ϱ_i, eigen vectors u_i of eigen decomposition of G_1 remain the same. □

3.3.3 Sequential Robust Randomized Gaussian Process Regression

As the data arrives continuously in an online setting, it is necessary to devise an intelligent way to update the kernel matrix to ensure that the computational power of the 5G-enabled Tactile Internet remote robotic surgery system is not exceeded, and real-time constraints are not violated. Therefore, we further developed a sequential robust randomized Gaussian Process Regression to update the kernel function. We update U and Σ using Algorithm 1 or Algorithm 2, which consider delayed/lost data in the training set. The approach is to use the eigen decomposition from the previous iterations G_j to compute the corresponding decomposition in the next iteration G_{j+1}. Given the initial kernel matrix as $G_j \approx U_j \Sigma_j U_j^T$, the covariance matrix of $\phi_{j+1}^{(d_{knot})}$ with itself and training as $G(\phi_{j+1}^{(d)}, \phi_{j+1}^{(d*)})$ and $G\left(\phi_{j+1}^{(d_{knot}*)}, \phi_{j+1}^{(d_{knot}*)}\right) + \sigma_n^2 I$ respectively, G_{j+1} is computed as follows:

$$G_{j+1} := \begin{bmatrix} G\left(\phi_j^{(d_{knot})}, \phi_j^{(d_{knot})}\right) + \sigma_n^2 I & G\left(\phi_{j+1}^{(d_{knot})}, \phi_{j+1}^{(d_{knot}*)}\right) \\ G\left(\phi_{j+1}^{(d_{knot}*)}, \phi_{j+1}^{(d_{knot})}\right) & G\left(\phi_{j+1}^{(d_{knot}*)}, \phi_{j+1}^{(d_{knot}*)}\right) + \sigma_n^2 I \end{bmatrix}. \tag{46}$$

Algorithm 3 Sequential Robust Randomized Gaussian Process Regression

```
1: Input: For $j = 1$, compute the kernel matrix $G_1$ using a kernel function and the intial input $\phi_1^{(d_{knot})}$
2: Compute $U_1$, $\Sigma_1$ using Algorithm 1
3: for $j \in \{2, 3, .....J\}$ do
4: Compute cross covariances $G(\phi^d, \phi^{d_{knot}*})$ and $G(\phi^{d_{knot}*}, \phi^{d_{knot}*}) + \sigma_n^2 I$
5: Form
   $$\bar{G}_j := \begin{bmatrix} U_{j-1}\Sigma_{j-1}U_{j-1}^T & G\left(\phi_j^{(d_{knot})}, \phi_j^{(d_{knot}*)}\right) \\ G\left(\phi_j^{(d_{knot}*)}, \phi_j^{(d_{knot})}\right) & G\left(\phi_j^{(d_{knot}*)}, \phi_j^{(d_{knot}*)}\right) + \sigma_n^2 I \end{bmatrix}$$
6: Using Algorithm 1 or 2, compute $U_j$, $\Sigma_j$
7: Compute $\vartheta^{-1}$ by Eq. (22)
8: Predict outputs for the new input $\phi_j^{(d_{knot})}$ using Eqs. (20) and (21)
9: end for
```

Table 3.2 Parameter settings and default values.

Simulation Parameters	**Values**
Environment	Intel (R) Core i7-7700 CPU @3.6GHz
Minimum contention Window	16
Maximum backoff	6
Empty slot duration	9 μs
SIFS	16 μs
DIFS	34 μs
PHY header	20 μs
MAC header	36 Bytes
RTS	20 Bytes
CTS	14 Bytes
ACK	14 Bytes
FCS	4 Bytes

To compute $G_{j+1} = \bar{U}_{j+1}\bar{\Sigma}_{j+1}\bar{U}_{j+1}^T$, we use the following matrix:

$$\bar{G}_{j+1} := \begin{bmatrix} U_j\Sigma_j U_j^T & G\left(\phi_{j+1}^{(d_{knot})}, \phi_{j+1}^{(d_{knot}*)}\right) \\ G\left(\phi_{j+1}^{(d_{knot}*)}, \phi_{j+1}^{(d_{knot})}\right) & G\left(\phi_{j+1}^{(d_{knot}*)}, \phi_{j+1}^{(d_{knot}*)}\right) + \sigma_n^2 I \end{bmatrix}. \tag{47}$$

A summary of the procedure is presented in Algorithm 5.

3.4 Simulations and Results on Real-World Dataset

In this section, we present our simulation results to illustrate the potential use of the various algorithms to support latency-sensitive H2M applications. We evaluated the streaming sparse Gaussian Process Regression (SparseGPR) [16], the proposed ℓ_1-SRLSMF Gaussian Process Regression-(GPR-ℓ_1) [1], the proposed

[1]We applied Theorem 1 in updating the covariance matrix of structured low-rank approximation in Gaussian Process Regression in [27]

SRLSMF Gaussian Process Regression (DirectGPR), LWPR, and PCA Gaussian Process Regression (GPR-ℓ_2) [35].

3.4.1 Metrics

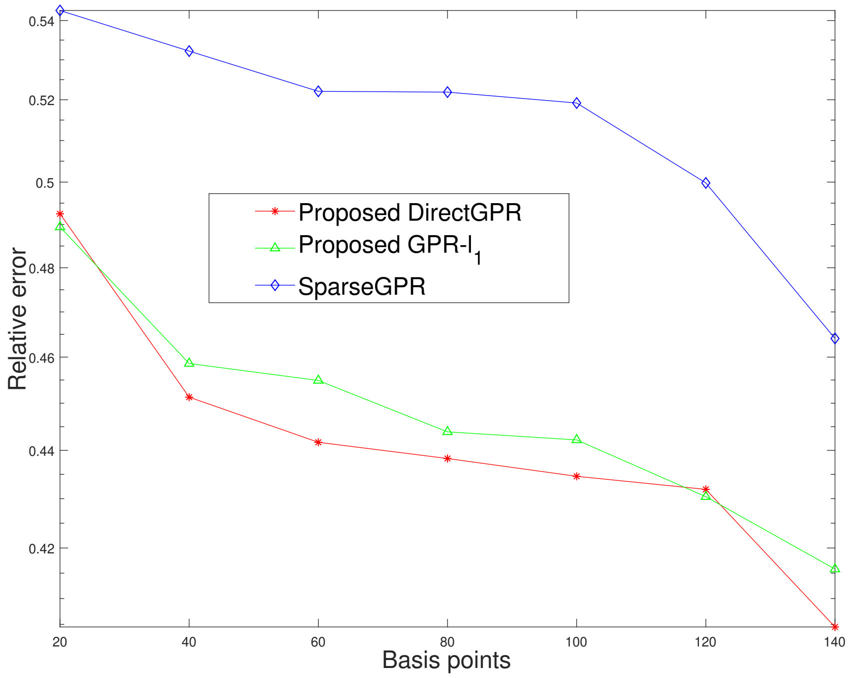
We considered two metrics to evaluate the various algorithms in 5G-enabled Tactile Internet remote robotic surgery.

1. *Relative error*: We use the relative error to examine the generalization capability of the methods under consideration by comparing the sent and received data during remote robotic surgery in 5G-enabled Tactile Internet networking infrastructure. In our considered scenarios, the relative error is computed as follows:
$$\text{Relative error} = \frac{||\hat{\xi}_m^d - \xi_m^d||}{||\xi_m^d||} \tag{48}$$

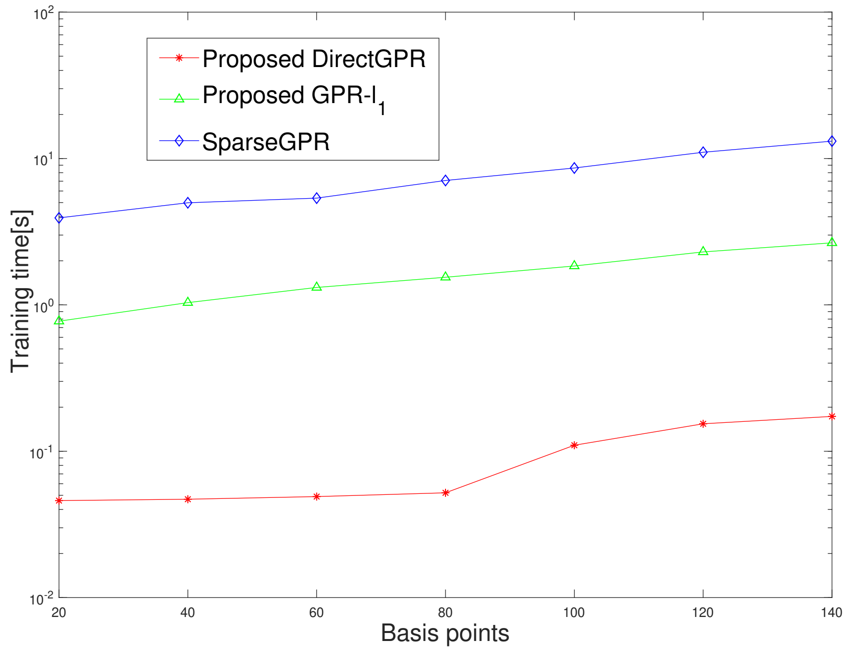
2. *Training and prediction time*: The training and prediction times of the methods under evaluation were considered to determine their adaptability in predicting delayed and/or lost data when performing the knot-tying gestures while considering the 1 ms deadline of remote robotic surgery in 5G-enabled Tactile Internet.

3.4.2 Robotic Surgery Dataset

We ran our simulations using the JIGSAW dataset [4], created by a cooperative project between Johns Hopkins University and Intuitive Surgical, Inc. using the Da Vinci surgical system. The JIGSAW dataset contains the kinematics of eight subjects. The dataset consists of three surgical tasks: suturing, needle passing, and knot-tying, which were collected from eight human subjects. Our goal is to predict the knot-tying gesture kinematic data that do not meet the 1 ms deadline during a remote robotic surgery. The dataset is comprised of time series of 76 kinematic data bytes that contains the numeric variables of four manipulators: master/slave and left/right manipulators. Each manipulator has 19 kinematic variables, comprised of the variables for the 3-variable Cartesian position, a 9-variable rotation matrix, the 3-variable linear velocity, the 3-variable angular velocity, and a gripper angle. These were collected at a sampling rate of 30 Hz. The kinematic dataset contains detailed information on position, velocity, orientation, etc.



(a) Relative error vs. basis points.



(b) Training time vs. basis points.

Figure 3.5 (a) Relative error and (b) training time vs. basis points for gesture 12 dataset.

3.4.3 Trace-based Simulation Results

We compare how well the variants of the low-rank matrix predict the next action of a surgeon by evaluating the relative error vs basis points. The SparseGPR [16] learns the hyperparameter and pseudo inputs in the same joint optimization while the proposed DirectGPR and GPR-ℓ_1 obtain hyperparameters from full GPR and the hyperparameters are fixed. We randomly split the dataset 10 times, each time with 90% of the points as the training set and the remaining 10% as the test set each time.The relative error for the various methods decreases almost linearly with increasing basis points, as shown in Figure 3.5a. The proposed DirectGPR outperforms the proposed GPR-ℓ_1 and SparseGPR [16] as shown in Figure 3.5a. The proposed DirectGPR achieves such a high accuracy because it learns the basis points appropriately to control the smoothness property of its fit, which contributed to the quality of its solution. We first compute the rank (L) using convex relaxed low-rank approximation and then used it to initialize nonconvex low-rank approximation. Hence, the proposed DirectGPR is more robust because we can obtain almost the exact rank of the matrix. This error reduction property ensures that, in practice, the iterations converge to a reasonable solution. Moreover, the proposed DirectGPR solves the underestimation problem of the proposed GPR-ℓ_1. SparseGPR is less accurate in Figure 3.5a because learning pseudo inputs by evidence maximization results in nonlinear optimization problem in which the solution often gets stuck in local optima.

Next, we compared the training time of the algorithms under a varying number of basis points, as depicted in Figure 3.5b. As the basis points increase, the training times increase almost linearly, as shown in Figure 3.5b. Generally, the proposed DirectGPR has a computational advantage over the SparseGPR. However, it should be noted that the training time of SparseGPR includes the time taken to infer the hyperparameters. In addition, SparseGPR takes a longer time to converge as it often gets stuck in local optima. The proposed DirectGPR outperforms the proposed GPR-ℓ_1, which we attribute to the fact that using least square minimization with regularized least square initialization to fit the data is faster than using regularized least square minimization alone under the same stopping criteria.

3.4.4 Online vs Offline Predictors

In the following, we evaluate the efficiency of our proposed online prediction against the offline prediction schemes. Figure 3.6 illustrates the relative error vs. time for the proposed scaled DirectGPR online predictor, the offline scaled DirectGPR predictor, and the LSTM-based offline predictor in both in- and out-of-sample testing on our robotic surgery dataset. The offline DirectGPR does not employ incremental eigen decomposition, while online DirectGPR uses the incremental eigen decomposition to update its training values. We

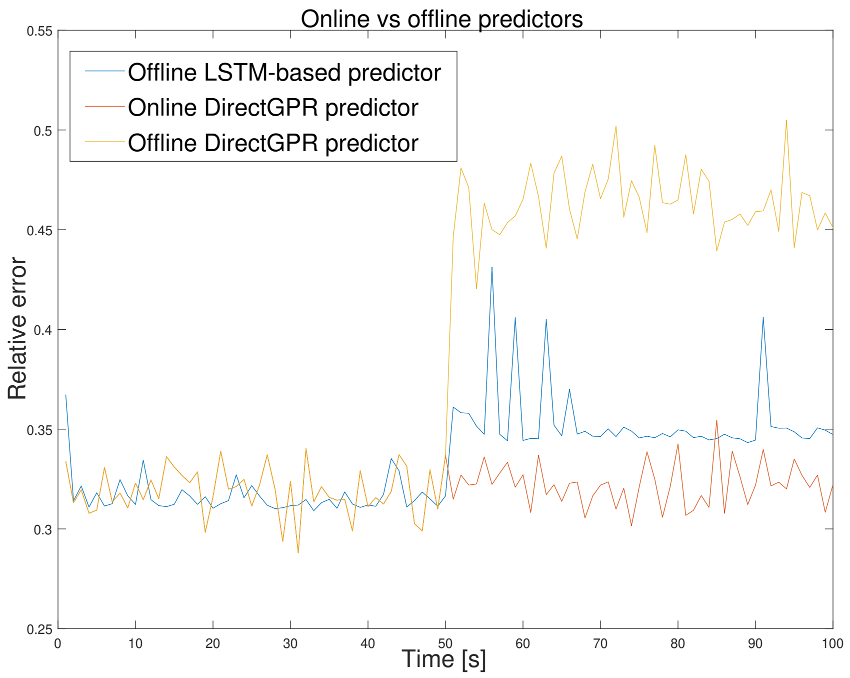


Figure 3.6 Relative prediction error vs. time for proposed online DirectGPR, offline DirectGPR and offline LSTM-based predictors.

randomly split each dataset 10 times; each time 90% of the dataset was used as the training set and the remaining data used as the test set. The relative error was averaged over 10 runs for each method. As shown in Fig. 3.6, for the case of in-sample testing on the suturing dataset, the performance of the offline LSTM-based predictor is comparable to our proposed online DirectGPR predictor and the offline DirectGPR for t<50 s. At some point, we observe that the accuracy of the LSTM-based predictor is better than that of our proposed DirectGPR and offline DirectGPR. This occurs because the proposed scaled online DirectGPR and offline DirectGPR trade prediction accuracy for speed. For $t > 50$ s, when the human subject's task shifts from suturing to needle passing, the relative error of both offline predictors (i.e., the offline LSTM-based predictor and the offline DirectGPR) increases. In contrast, our proposed online DirectGPR predictor retains similar level of accuracy, even when such a shift occurs, because our online scaled DirectGPR predictor can deal with out-of-sample prediction better than dense offline LSTM-based and offline DirectGPR predictors for the unseen needle passing dataset. More specifically, unlike the dense LSTM offline predictor and offline DirectGPR, our scaled online model is able to update its model parameters in an online manner using incremental eigen decomposition to improve its prediction accuracy. It is worth mentioning that the dense LSTM-based predictor can handle out-of-sample prediction better than the offline scaled DirectGPR because it is not a compressed model.

3.4.5 Simulation of Real-World Robotic Surgery Dataset with Tactile Internet Networking Infrastructure

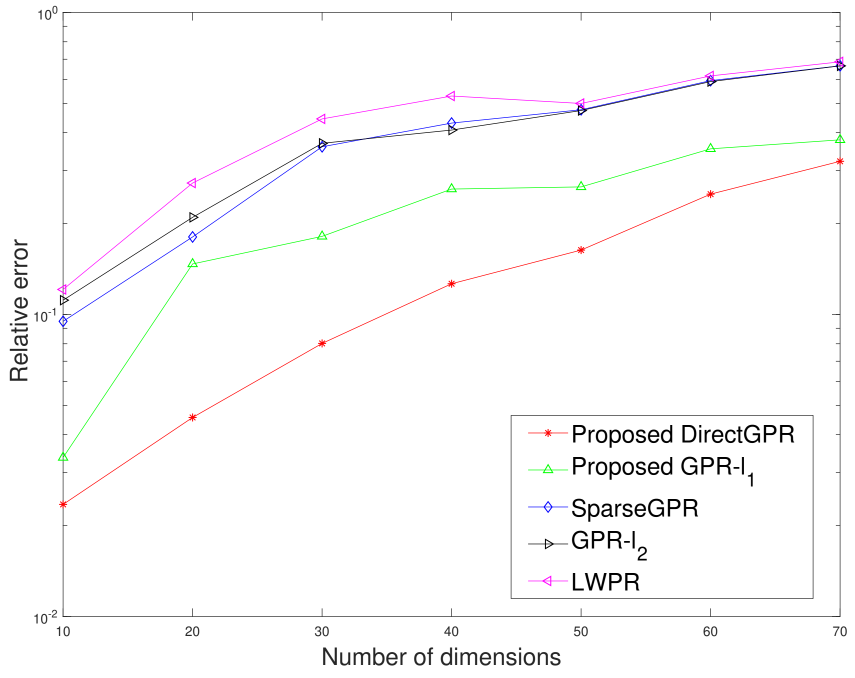
Although our proposed framework is applicable to any Tactile Internet networking infrastructure, we consider a fiber-wireless (FiWi) enhanced Long Term Evolution Advanced (LTE-A) Heterogenous Network (HetNets) in [88] in our simulations and evaluate the performance of the proposed framework. We chose FiWi enhanced LTE-A HetNets for the following reasons:

1. FiWi access networks combine the coverage-centric LTE-A HetNet with capacity-centric WiFi access networks to offer low-cost, high-speed mobile offloading, that is achieved through a high capacity next-generation passive optical network (PON) and Gigabit-class wireless local area networks (WLANs) that are known to operate at data rates 100 times higher than those of cellular networks. FiWi access networks rely on heterogeneous network architecture to improve capacity, a multitude of small cells to improve coverage, and carrier aggregation to achieve high data rates. Such networks have proven to achieve almost 100-fold gains in area capacity and 10 Gbps peak data rates [64].

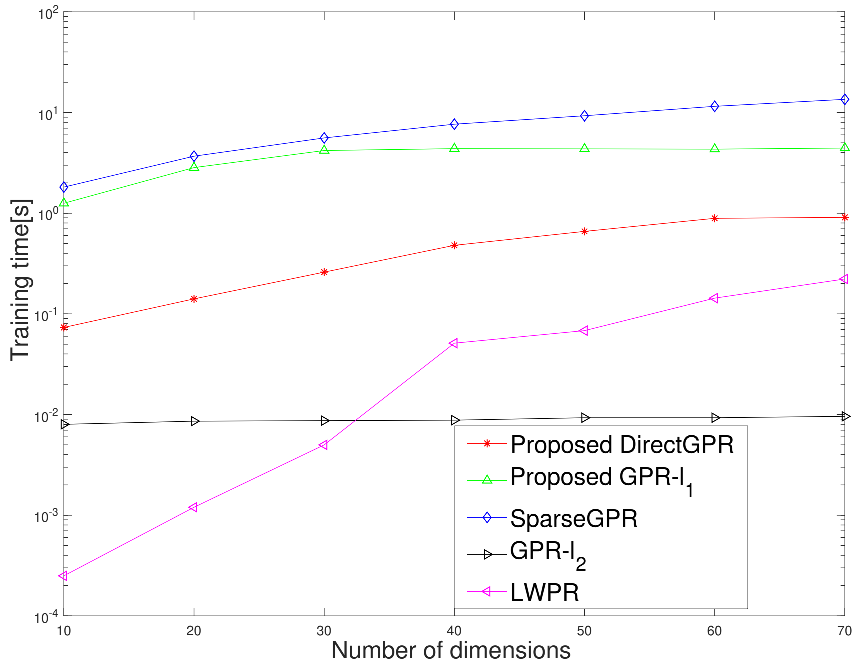
2. FiWi enhanced mobile networks offer distributed storage and processing capabilities, which can be exploited to offer an immersive teleoperation experience for human subjects. Computing and storage resources such as cloudlets and fog nodes are intentionally placed closer to the end-users to offer low latency, low jitter, low data transfer cost, and scalability [89].

In the backhaul, we considered a time or wavelength division multiplexing IEEE 802.3ah/av 1 Gb/s Ethernet passive optical network with a fiber length of 20 km between the central optical line terminal (OLT) and the remote optical network unit (ONU)/access point (AP) that connects a surgeon domain with a patient domain [64]. Both the surgeon console and patient-side robot may connect to a cellular network base station (BS) or an IEEE 802.11n/ac/s WLAN mesh portal point (MPP). The ONU-BS/MPP with artificial intelligence enhanced multi-access edge computing (MEC) servers are collocated at the patient's optical-wireless interface in order to perform the prediction of delayed and/or lost haptic commands. We apply the RTS/CTS, a control message of the distributed coordination function (DCF) access mechanism of IEEE 802.11, at the nodes (patient side robot and surgeon console). Both the patient side robot and the surgeon console operate at 56 Mbps. Our experiment had the surgeon on a single communication channel with no competing users to reduce the impact of delay and guarantee safety. Table 3.2 summarizes our parameter settings and default values.

To verify the robustness of the proposed framework compared to other incremental algorithms in terms



(a) Relative error of the various algorithms vs number of dimensions.



(b) Training time of the various algorithms vs number of dimensions.

Figure 3.7 Regression and Training time results on Gesture 12 dataset under varying dimensions

of delay and packet loss, we train the various algorithm on the Gesture 12 knot tying dataset under varying delay and packet loss online in a Tactile Internet environment. We evaluate the performance of our proposed Direct GPR and GPR-ℓ_1 methods in term of relative error compared to incremental learning methods LWPR, SparseGPR, and GPR-ℓ_2. Figure 3.7a depicts the relative error vs. the number of dimensions for Gesture 12 of the knot-tying dataset. As the number of dimensions increase, the relative errors also increases almost linearly for all method under consideration, because increasing the number of dimensions means there are more data points to predict. Further, we observe that the proposed DirectGPR algorithm slightly outperforms the proposed GPR-ℓ_1 because of the regularization technique it uses to fit the data. The proposed DirectGPR achieves such a high accuracy because of the its well designed initialization and because it incorporates the delay/packet loss condition into the model to ensure robust model fitting. The low delay/packet loss means that the proposed DirectGPR method can overcome the few instances of delayed/lost data in the training dataset to predict the surgeon's next action in a reliable manner, as shown in Figure 3.7a. SparseGPR, GPR-ℓ_2, and LWPR method achieve poor results because the corrupted data are present within the kernel matrix and distort the kernel matrix inversion. In addition, Sparsity-based GPR methods such as the proposed DirectGPR, proposed GPR-ℓ_1, SparseGPR, and GPR-ℓ_2 outperform LWPR on the small amount of gesture 12 datasets because they are easier to train. LWPR requires appropriate clustering of the state space, not a straightforward process on any dataset.

Next, we evaluate the training time of various algorithms. Fig. 3.7b depicts the training time vs. number of dimensions for different algorithms. We observe that the training time of the proposed DirectGPR algorithm is smaller than that of the proposed GPR-ℓ_1 algorithm. The reason is that the DirectGPR algorithm ignores the lost data, as opposed to the GPR-ℓ_1 algorithm, which resort to regularization to recover the data. On the other hand, the LWPR algorithm achieves the smallest training time when the number of dimensions is below 30. The training time of LWPR gets worse with increasing dimensions. The timing behavior of LWPR could be refined by adjusting the threshold for the creation and removal of receptive fields. However, tuning in LWPR is inefficient and will likely deteriorate the generalization performance. While SparseGPR does not recover the lost data, the pseudo input location selection and hyperparameter optimization are performed together, thereby having the highest training time. Given that the GPR-ℓ_2 algorithm does not perform any data recovery and so it has the smallest training time between 30 and 70 dimensions.

Next, we compare our proposed algorithms in the presence of packet loss rate for G12 gesture datasets. Figure 3.8 shows the relative error vs. the packet loss rate. We observe from Fig. 3.8 that for both DirectGPR and GPR-ℓ_1 the relative error grows with increasing packet loss rate. It is also evident from Fig. 3.8 that DirectGPR outperforms GPR-ℓ_1 for small packet loss rates (i.e., 5%-15%) as ignoring the data loss under the

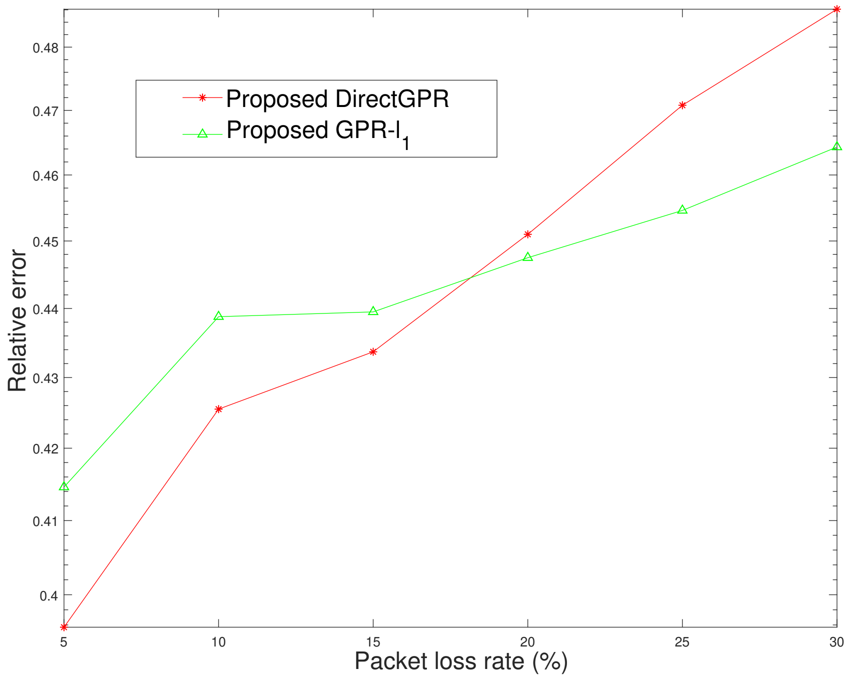
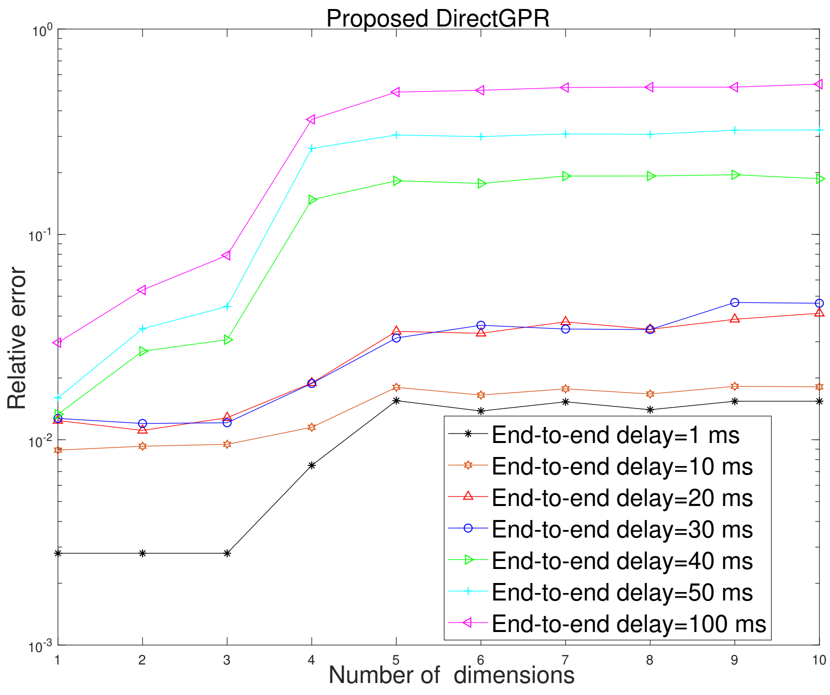


Figure 3.8 Relative error vs. packet loss rate for DirectGPR and GPR-ℓ_1 methods.

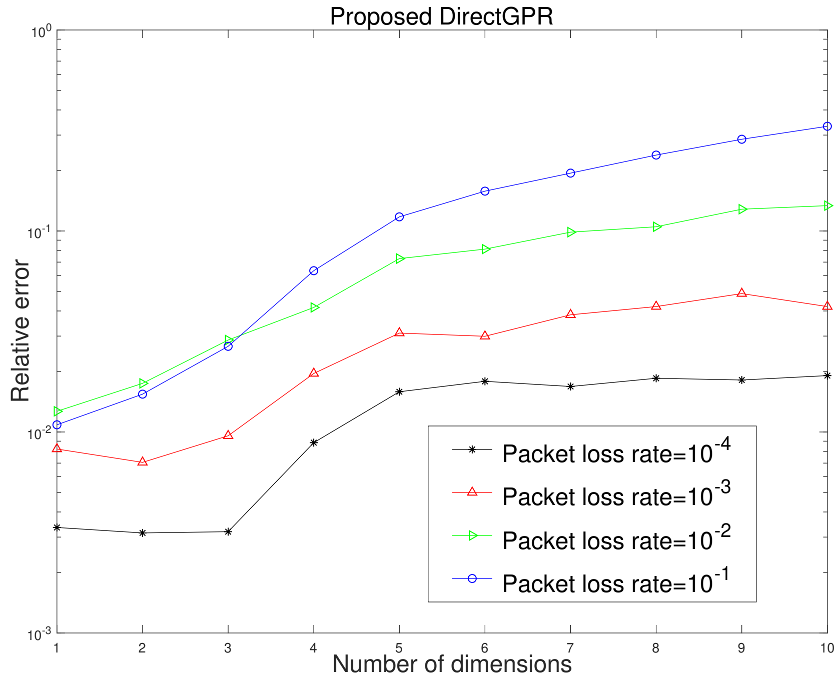
assumption that they are small is better than using a regularization scheme to deal with such losses. Furthermore, GPR-ℓ_1 performs better than DirectGPR at high packet loss rates (i.e., 15%-30%). This phenomenon can be attributed to the robustness of the GPR-ℓ_1 regularization scheme to fit the data at high packet loss rates.

In the following, we evaluate the performance of the proposed framework in the presence of delay and packet loss. We use haptic traces (i.e., G12 gesture datasets) of different surgical tasks. First, we incorporated packet loss rates of 10^{-4}, 10^{-3}, 10^{-2}, and 10^{-1} obtained after 50 haptic sessions into the original gesture 12 datasets to represent the estimated randomly lost haptic sample. Then, we initialized the original G12 gesture datasets with end-to-end delays of 1 ms, 10 ms, 20 ms, 30 ms, 40 ms, 50 ms, and 100 ms also obtained after 50 haptic sessions, to represent the estimated haptic samples within a particular time interval during transmission. We consider 1-ms-apart time-steps. Once the delay experienced by a haptic packet exceeds the given threshold, our proposed DirectGPR model provides the patient-side robot with the predicted haptic command.

During transmission, a surgeon manipulates the console to send a haptic sample to the patient side robot. The sent haptic sample is expected to arrive within the 1 ms targeted delay. However, if the patient side robot fails to receive the sample in time, the edge predictor (i.e., the proposed DirectGPR) performs a single- or multi-step-ahead prediction depending on the latency. Figure 3.9a illustrates the relative error versus number

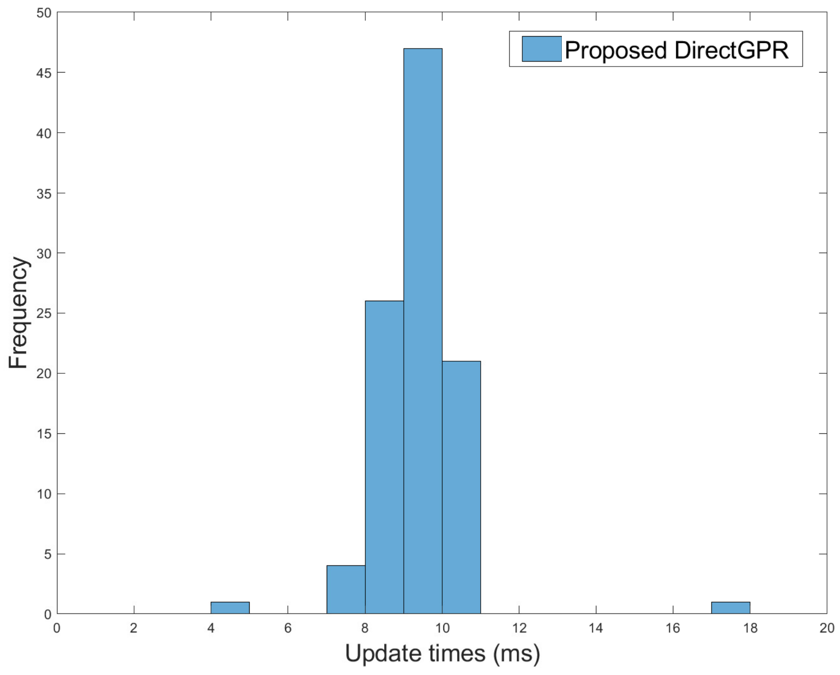


(a) Relative error vs. number of dimension for end-to-end delay initializations of 1 ms , 10 ms, 20 ms , 30 ms , 40 ms, 50 ms, and 100 ms using the proposed DirectGPR.

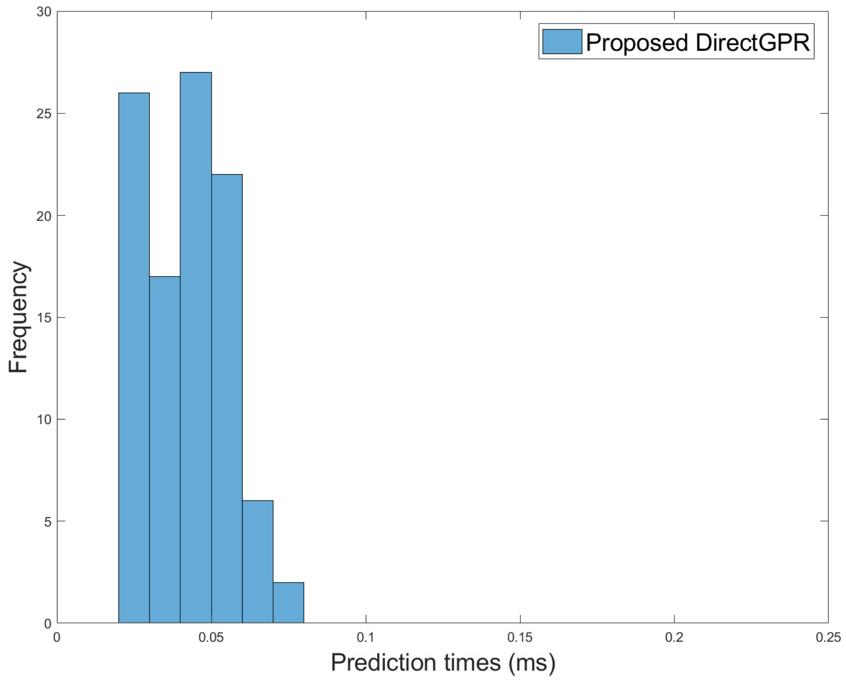


(b) Relative error vs. number of dimensions for different packet loss rate of 10^{-4}, 10^{-3}, 10^{-2}, and 10^{-1} using the proposed DirectGPR.

Figure 3.9 Relative error vs. number of dimensions for different values of (a) end-to-end latency and (b) packet loss rate using the proposed DirectGPR.



(a)



(b)

Figure 3.10 Histogram of (a) update and (b) prediction times (in milliseconds) of the proposed DirectGPR on the patient side during retraining and prediction phases, respectively.

of dimensions for different values of the end-to-end latency, and Fig. 3.9b shows the relative error versus number of dimensions for different values of packet loss rate. We observe from Figs. 3.9a and 3.9b that for the case of fixed dimensions (1-3), the proposed scheme achieves a relative error within the range of $10^{-3} - 10^{-2}$ for the end-to-end delay of 1-10 ms and a packet loss rate of $10^{-4} - 10^{-3}$. We note the the obtained relative error by our proposed method falls below the so-called just noticeable different (JND), which, according to the well-known Weber's law, is the minimum change in the magnitude of a stimulus that can be detected by humans. The JND threshold for force perception with hand and arm is $7\pm1\%$ [90]. The high reliability achieved at low dimension, low packet loss, and low end-to-end delay can be attributed to the convex relaxation approach used as the initialization for our proposed framework, allowing convergence to a good solution by fine-tuning the model parameters, as well as, error resilience of the proposed DirectGPR. Another interesting observation is that, as the packet loss rate and end–to–end delay grows for a given number of dimensions, the prediction performance of the proposed framework sometimes deteriorates as seen in Fig. 3.9a and 3.9b. This is expected since the proposed DirectGPR algorithm does not impute the missing data but rather ignores it under the assumption that the lost data is very small. Therefore, in the case of high packet loss rate and end-to-end delay, imputation of the missing data is required to further improve the prediction accuracy of the proposed DirectGPR. It is possible to achieve high prediction reliability (i.e., 10^{-3}-10^{-2}) for the case of low packet loss rate (i.e., $10^{-4} - 10^{-3}$), low end-to-end delay (i.e., 1-10 ms), and a low dimension of gesture datasets (i.e., 1-3). The achieved reliability is less than the expected reliability of 99.999% even after fine tuning the model parameters and recovering the lost data. This lowered reliability can be attributed to the inherent packet loss rate and end-to-end delay due to communication malfunction and hardware impairment of the da Vinci robotic surgery system used in gathering the dataset [4], which compounded the lost data. To further improve reliability, the use of network slicing and advanced edge traffic management techniques are suggested.

In the following, we examine the update and prediction times of our proposed methods on gesture 12 3-dimension datasets of a human subject performing a knot-tying task. Our results were obtained in a local PC with Intel (R) Core i7-7700 CPU @3.6GHz. Histogram of model update times are illustrated in Figure 3.10a, where we observe that the update times of the model are centered around 10 ms on the gesture 12 3-dimension datasets. The y-axis (i.e., the frequency) in Figs. 3.10a and 3.10b represents the number of occurrences of a prediction and/or update time of our proposed DirectGPR while predicting delayed/lost data in remote robotic surgery. It should be noted that the haptic session was non-local, but the prediction by trained DirectGPR was done locally in the patient domain. The histogram of the prediction and the update times were obtained based on an average of 50 interaction haptic sessions between the human subject domain and the patient domain during which the proposed DirectGPR performed predictions of the delayed/lost data. These longer update times occurred because updating/re-training the model parameters is a computationally intensive task. To reduce these times, we could incorporate our proposed DirectGPR into a federated or distributed learning scheme that can be run on a more powerful set of edge servers in practical scenarios. The histogram of the prediction times of the proposed DirectGPR is presented in Fig. 3.10b, and indicates that the proposed DirectGPR can make predictions that are well under 1 ms. This advantageous performance can be attributed to the exploitation of the scalability of low-rank matrix factorization.

3.5 Conclusion

In this chapter of the thesis, to offer an immersive and transparent teleoperation experience for a surgeon, we developed an efficient and lightweight framework for predicting haptic commands in a remote knot-tying task for 5G-enabled Tactile Internet remote robotic surgery. Numerical simulation results show that the proposed model offers a good trade-off between performance and computational costs. Specifically, the proposed DirectGPR optimizes a smooth and nonconvex proxy for the original cost function by alternating between two steps: convex relaxation initialization in the first step followed by nonconvex optimization in the second step. With a fixed hyperparameter obtained from a full Gaussian Process Regression, the proposed framework utilizes previous decomposition results to compute the next decomposition result after new data is appended. The proposed DirectGPR gave a sparser model than the proposed GPR-ℓ_1 and SparseGPR. Therefore, the proposed DirectGPR reduced the underestimation problem of the proposed GPR-ℓ_1 and SparseGPR.

On a data lost recovery experiment in a Tactile Internet environment, the proposed DirectGPR predictive performance exceeds that of the proposed GPR-l_1 because the proposed DirectGPR incorporates the network conditions of 5G-enabled Tactile Internet, which makes it both lightweight and more robust to delayed/lost data than the proposed GPR-ℓ_1, which uses a regularization technique to compensate for lost data. Both the proposed DirectGPR and GPR-ℓ_1 outperformed SparseGPR, LWPR, and GPR-ℓ_2 , none of which recover lost data. In addition, LWPR performed the worse because it is not able find the appropriate balance between the creation and removal of receptive fields to improve it prediction performance.

Chapter 4

Enhanced Stochastic Gradient Descent Algorithm for Machine Learning Training and Its Application in Remote Surgical Gesture Recognition

4.1 Introduction

Remote robotic surgery enhances patients' quality of experience by guaranteeing shorter hospital stays, minimizing bleeding and incision size, lowering the risk of infection, and expediting recovery. [20]. In remote robotic surgery, the surgical process can be decomposed into predefined units called surgeme or gesture [12] as shown in Table 4.1. Surgemes serve as the foundational building block in understanding robotic surgery. Surgical gestures can be recognized from videos and kinematic datasets [4]. The ability to recognize surgical gestures from video and the kinematic dataset is vital for automating surgical activities like surgical gesture recognition. The kinematic dataset is made up of the robot's movement design and its movement in space whiles robot video dataset represents the whole operating room. Both datasets, provide essential knowledge about the interaction between surgical tools and body tissue. Furthermore, they provide useful discriminating features that can serve as an elementary unit for surgical data science (i.e., surgical gesture recognition) [53, 91].

Encouraged by computer vision, pattern recognition and machine learning translation, surgical gesture recognition has become a rapidly thriving field [91]. Unlike computer vision, pattern recognition and machine translation, surgical gesture recognition datasets are limited [4]. Nevertheless, a commonly available dataset, JHU-ISI gesture and skill assessment working set [4], has made the study of surgical gesture recognition and classification [53] easy. Many surgical data scientists depend on the JIGSAW dataset for developing their automated recognition and classification algorithms [91]. JIGSAW dataset has both video and kinematic datasets. The video and kinematic dataset both present the same discriminate ability when the right features are used for training the model [91].

Recognizing surgical gestures is a complicated task due to the dataset's lack of variability and fine-grained structure [91]. Several machine learning techniques have been used for surgical gesture recognition, namely hidden markov model and Markov/Semi-markov conditional random field (MsM-CRF) [91] to deal with this issue. The hidden Markov-based models for surgical gesture recognition have short term temporal dependencies of the surgical motion [91]. Lately, deep learning has attracted a lot of attention as a main force in surgical gesture recognition domain because deep learning models can learn long term temporal dependencies of surgical motion [91]. Nevertheless, deep learning models are crippled by poor convergence rate of the model due to the several nonlinear functions in the model.

Convergence of the surgical gesture recognition model is important for increasing patient quality of experience. If the model does not converge to a good solution, the gesture recognition model will have poor accuracy leading to wrong estimation of the surgical gesture. Such wrong estimation of the surgical gesture can be life threatening in remote robotic surgery. Thus, we have decided to enhance the learning phase of surgical gesture recognition to boost their convergence rate. Several researchers have attempted to settle this problem using dropout [92], optimizers [93], data-augmentation [94], batch normalization [95] and efficient model compression techniques [25] for optimizing networks for recognition applications. Yet, convergence rate of these machine learning models is only almost optimal. This suggests that while the machine learning models are approaching their best possible performance in terms of how quickly they reach an optimal solution, they are not quite achieving the ideal convergence rate. In other words, they are close to being as efficient as possible, but there is still some room for improvement.

Adaptive SGD enjoys fast convergence and works effectively in optimizing many networks for recognition applications [39]. Nevetheless, it is observed that ASGD does not function well when the dataset under consideration has a combination of large and small coordinate values [28]. For this reason, in this chapter, we focus on building a new adaptive stochastic gradient descent optimizer to address this issue to boost the convergence rate of surgical gesture recognition. We propose to exploit a non-uniform p-norm-based concept [96] to build an ASGD to train machine learning model for surgical gesture recognition with high convergence rate. To be more specific, we fix the small learning rate dilemma problem [28] associated with ASGD by setting a threshold to divide the system into large and small categories, and each category is given different system requirements [96]. The proposed ASGD can achieve fast convergence rate and good generalization performance by setting a base learning rate according to the system requirement. Our approach is fundamentally different from adagrad's [97] update rule since adagrad [97] updates frequently occurring features with low learning rates and infrequently occurring features with high learning rate.

The remainder of the chapter is organized as follows: machine learning solution for surgical task is described. Then, experiments on image processing and remote surgical gesture recognition is provided. Finally, chapter is concluded. Provided in Table 3 is the abbreviation list.

4.2 A Machine Learning Approach for Surgical Gesture Recognition

The objective of this chapter is to construct a framework for surgical gesture recognition in remote robotic surgery. The problem definition is presented, followed by background information on 2D CNN and Convolutional LSTM models. Then, the proposed adaptive method for enhancing the convergence rate of the machine learning models and application to remote robotic surgery are discussed. Also, the convergence analysis of the Proposed ASGD in convex and nonconvex settings are presented.

4.2.1 Problem Definition

The main problem we are trying to address is predicting the gestures of a surgical task. We formulate the video-based gesture recognition problem as follows. We would like to predict the first gesture $y_1 \in Y$ which was executed at time $t = 1$ for each $t = 1, ..., T$, where $y = \{1, ...Y\}$. We denote i as the index of the sample, t as the iteration, d as the number of video parameters, $\mathbf{x}_i$ as d-dimensional video features of surgical gesture recognition and y_i as a gesture of a surgical task as the labels.

4.2.2 Background on 2D CNN and Convolutional LSTM model for Gesture Classification

We formulate the problem as gesture recognition problem, where an 2D CNN and LSTM model are trained to predict the gesture of surgical task [53].

Table 4.1 Gesture Vocabulary [4].

Abbreviation	Surgical gesture discription
Suturing	y_1: Reaching for needle with right hand y_2: Positioning needle y_3: Pushing needle through tissue y_4: Transferring needle from left to right y_5:Moving to center with needle in grip y_6:Pulling suture with left hand y_8:Orienting needle y_9: Using right hand to help tighten suture y_{10}: Loosening more suture y_{11}: Dropping suture and moving to end points

Following the methodology outlined in [53], albeit with with slight adjustments, we employ a Convolutional LSTM network architecture for the gesture classification task. Each input frame undergoes five passes through a 2D convolutional layer, accompanied by a max pooling layer after each convolutional operation. Subsequently, the outputs from these convolutional layers are flattened before being inputted into the LSTM with 16 layers. The last output of the LSTM block is then forwarded into a dense layer comprising 8 nodes. Finally, another dense layer with 10 nodes is utilized to generate the classification results, representing the 10 surgemes defined in the JIGSAWS suturing dataset [53]. Details of the LSTM models are as follows.

Let c_i, i_i, f_i, and o_i denote the cell state, input state, forget gate, and output gate of an incoming packet i, respectively. Let W_c, W_{INPUT}, W_f, and W_o denote the weight matrix of the cell state, input gate, forget gate, and output gate of an incoming packet i, respectively. Let b_c, b_i, b_f, and b_o denote the bias vector of the cell state, input gate, forget gate, and output gate, respectively [98]. The cell state records the recent state. Input gate is the amount of the input of an incoming packet i is saved. Forget gate aids the network in forgetting past input information and resets the memory cells [99].

the general formulation for gate of LSTM was defined by Shi et al. [98] as

$$g = \sigma(W.[\mathbf{x}_i, y_{i-1}] + b). \tag{49}$$

Given sigmoid activation function $\sigma(\mathbf{x}_i) = 1/(1 + e^{-\mathbf{x}_i})$, the cell state c_i which is the addition of the output of input gate and tanh layer is given below

$$c_i = f_i.c_{i-1} + i_i.\tanh(W_c.[\mathbf{x}_i, y_{i-1}] + b_c). \tag{50}$$

The selection of the activation function can be based on the gating mechanism of the neural network model under consideration [100]. The activation functions most frequently utilized include the rectified linear unit (ReLU) and sigmoid [100]. Compared to sigmoid activation function, ReLU activation can handle vanishing gradient problem in other deep neural networks but diverges on LSTM. Therefore, ReLU activation function does not perform well empirically on LSTM. When sigmoid activation is used with LSTM, vanishing gradient problem is mitigated by the network structure of LSTM [100]. Therefore, sigmoid activation function is the preferred choice for most LSTM models [98]. Tripathi et al. [98] defined the following steps for computing the forget gate, output gate and output of LSTM unit for the classification problem. At step i, the input gate of LSTM unit is computed as

$$i_i = \sigma(W_{INPUT}.[\mathbf{x}_i, y_{i-1}] + b_i). \tag{51}$$

At step i, the forget gate of LSTM unit [98] is computed as

$$f_i = \sigma(W_f.[\mathbf{x}_i, y_{i-1}] + b_f). \tag{52}$$

At step i, the output gate of LSTM unit [98] is computed as

$$o_i = \sigma(W_o.[\mathbf{x}_i, y_{i-1}] + b_o). \tag{53}$$

At step i, the output of LSTM unit [98] is computed as

$$\mathbf{y}_i = o_i.\text{tanh}(c_i). \tag{54}$$

4.2.3 The Proposed Adaptive Method for Enhancing the Convergence Rate of ASGD

The main contribution of this chapter is to improve the accuracy of adaptive SGD by adapting the stepsize of the model in the remote robotic surgery system. Taking inspiration from p-norm adaptive filtering algorithms [96, 101–103], we propose to improve the step-size selection scheme in ASGD to promote fast convergence for our robotic surgery model. In particular, given the gradient $z_t = \bigtriangledown g(W_t)$, the adaptive parameter β, the momentum $n_t = \beta_1^t n_{t-1} + (1 - \beta_1^t) z_t$, the preconditioner $H_t = \beta_2 H_{t-1} + (1 - \beta_2) z_t^2$, and small constant $0 < \epsilon << 1$, the adaptive gradient descent in [30] is given as

$$W_{t+1} = \left(W_t - \frac{\alpha}{\sqrt{(H_t + \epsilon)}} n_t \right). \tag{55}$$

We exploit the non-uniform p-norm concept [96] and introduce the large and small coordinate values idea into ADAM [30]. For small coordinates with expected value $[|(H_t)|] = \frac{1}{d}\sum_l h_l$, the update rule is [96]

$$f_{small} = \frac{sgn\left([|(H_t)|] - [H_t]\right) + 1}{2}. \tag{56}$$

and then, for large coordinates the update rule is [96]

$$f_{large} = \frac{sgn\left([H_t] - [|(H_t)|]\right) + 1}{2}. \tag{57}$$

The base learning rate is based on whether $h_{t,l}$ coordinate values are small or large. Given an auxiliary variable u and constant value C, the base learning rate is defined as a linear function for small and large coordinate values

$$\alpha_{base} = uf_{small}(H) + C \tag{58}$$

and

$$\alpha_{base} = uf_{large}(H) + C. \tag{59}$$

For learning the small and large base learning rate α_{min} and α_{max}, we defined a piece-wise function

$$\alpha_{base} = \begin{cases} \alpha_{small} & \text{if} \quad \text{H}_t \quad \text{is} \quad \text{small} \\ \alpha_{large} & \text{if} \quad \text{H}_t \quad \text{is} \quad \text{large} \end{cases} \tag{60}$$

If $h_{t,l}$ values are small, the effective learning rate $\frac{\alpha_{base}}{\sqrt{h_{t,l}}}$ will be large, and if $h_{t,l}$ values are large, the effective learning rate $\frac{\alpha_{base}}{\sqrt{h_{t,l}}}$ will be small [28, 42]. Therefore, we assigned a large base learning rate α_{max} when majority of the coordinate values are larger than the mean value and a small base learning rate α_{min} when majority of the coordinate values are smaller than the mean value. In summary, we can avoid small learning rate dilemmas [28,42] in both learning situations by selecting a good base learning rate that improves the model's empirical results of the surgical gesture recognition task.

The proposed ASGD update rules for small and large coordinate are

$$W_{t+1,small} = \left(W_t - \frac{\alpha_{min}}{\sqrt{(H_t + \epsilon)}} n_t\right) \tag{61}$$

and

$$W_{t+1,large} = \left(W_t - \frac{\alpha_{max}}{\sqrt{(H_t + \epsilon)}} n_t\right). \tag{62}$$

We can incorporate the improvised equations 56, 57 [96] and 59 into different ASGD methods to improve their performance compared to the standard ASGD methods for certain finite number of epochs (fast convergence).

4.2.4 Convergence Analysis of the Proposed ASGD in Convex Setting

The convergence of the proposed algorithm in convex setting is guaranteed by Theorem 2 below. The details of Theorem 2 follow AMSgrad [40], with the base learning rate α_{base} modified to $(uf(H) + C)$. We define the following assumption from [30] with the epsilon value ϵ dropped ,

Assumption 1. *[30] 1: let function $g : \Re^d \to \Re$ be convex, then $x, y \in \Re^d, g(x) \in \nabla g(x)^T(y - x)$, then*

$$g(y) \geq g(x) + \nabla g(x)^T(y - x). \tag{63}$$

Theorem 2. *: Given $\{W_t\}_1^T$ and $\{H_t\}_1^T$ are sequence generated by algorithm 1 and 2. Suppose $\alpha_t = \frac{u.f(H)+C}{\sqrt{(t)}}$, $\gamma \triangleq \frac{\beta_1^2}{\sqrt{\beta_2}}$ $\beta_{1,t} = \beta_1\lambda^{t-1}$, and $\lambda \in (0, 1)$, then if the set χ has a bounded diameter D_∞, i.e., $||W_t - W_{t'}||_\infty \leq D_\infty$ for all $W \in \chi$ and z_t has the bounded gradient, i.e., $||\nabla g_t(W)||_\infty \in Z_\infty$, we have the upper bound of the regret of the proposed algorithm as*

$$\begin{aligned} R_T \leq & \frac{D_\infty^2}{2(uf(H) + C)(1 - \beta_1)} \sum_{l=1}^{d} \sqrt{T} h_{T,l}^{\frac{1}{2}} \\ & + \frac{uf(H) + C(\beta_1 + 1)Z_\infty^{(1)} \times \sqrt{1 + \log T}}{(1 - \beta_1)(1 - \beta_2)^{\frac{1}{2}}(1 - \gamma)^2} \sum_{t=1}^{T} ||z_{1:T,l}||_2 + \\ & \sum_{l=1}^{d} \frac{D_\infty^2 Z_\infty^1 \beta_1}{(uf(H) + C)(1 - \beta_1)(1 - \lambda)^2}. \end{aligned} \tag{64}$$

*Then, we summarize the following lemma in the convex analysis needed to prove Theorem. **The subsequent steps closely resemble the proof outlined in Theorem 2 in AMSgrad [40].***

Lemma 1. *: This lemma follows the assumption in Theorem 2 [30, 40] and we have*

$$\sum_{t=1}^{T} \sum_{l=1}^{d} \frac{(u.f(H) + C).n_{t,l}^2}{(h_{t,l})^{\frac{1}{2}}} \leq \frac{(uf(H) + C)Z_\infty^{(1)}\sqrt{1 + logT}}{(1 - \beta_1)(1 - \gamma)(1 - \beta_2)^{\frac{1}{2}}} . \sum_{l=1}^{d} ||z_{1:T,l}||_2. \tag{65}$$

We would like to clarify that this lemma is a minor modification of the version of AMSgrad [40].

Lemma 2. *[30]: Suppose $H \in S_d^+$ is a symmetric positive definite matrix, $p \in (0, \frac{1}{2}]$, $a_1 = \prod_{\{W\}H^p}(b_1)$*

and $a_2 = \prod_{\{W\}H^p}(b_2)$*, then we have*

$$||H^{\frac{1}{2}}(a_1 - a_2)||_2 \leq ||H^{\frac{1}{2}}(b_1 - b_2)||_2. \tag{66}$$

Fundamentally, our method shown in Algorithms 4 and 5 is different from Adagrad's method [97]. Adagrad [97] updates the frequently occurring features with low learning rates and less frequently occurring features with high learning rates. In contrast, we define a small base learning rate when many of the coordinate values are smaller than the mean and a large base learning rate when a large portion of the coordinate values are larger than the mean.

4.2.5 Ergodic Convergence Analysis of the Proposed ASGD in Nonconvex Setting

In this subsection, we establish the convergence proof in a nonconvex setting by following the methodology outlined in Zhou et al. [32] and substituting the base learning rate with a linear function $uf(H) + C$.

Theorem 3. *Under the following assumptions:*

Assumption 2. *[32] For a differentiable function g, there exists a constant L such that* $||\nabla g(x) - \nabla g(y)|| \leq L|||x - y||$ *for all* x, y*, and g is lower bounded.*

and

Assumption 3. *[32]: The function* $g(W) = \mathbb{E}_\xi\, g(W; \xi)$ *has a* Z_∞*-bounded stochastic gradient, meaning that for any* ξ, $g(W; \xi) \leq Z_\infty$, $\beta_1 < \beta_2^{\frac{1}{2}}$, $\alpha_t = (u.f(H) + C)_t$*, and a sequence* $\{Q_i\}_{i=1}^3$ *where* $||z_{1:T,i}||_2 \leq Z_\infty$ *for* $t = 1, \ldots.., T$*,*

the iteration W_t *of the proposed ASGD satisfies the*

$$\frac{1}{T-1}\sum_{t=2}^{T}\mathbb{E}[|| \bigtriangledown g(W_t)||_2^2] \leq \frac{R_8}{T.(u.f(H)+C)} + \frac{R_9 d}{T} + \frac{(u.f(H)+C)R_{10}d}{T^{\frac{1}{2}}} \tag{67}$$

where

$$R_8 = 2Z_\infty \Delta g, \tag{68}$$

$$R_9 = \frac{2Z_\infty^3 \epsilon^{-\frac{1}{2}}}{1-\beta_1} + 2Z_\infty^2, \tag{69}$$

$$R_{10} = \frac{2LZ_{\infty}^2}{\epsilon^{\frac{1}{2}}(1-\beta_2)^{\frac{1}{2}}(1-\frac{\beta_1}{\beta_2^{\frac{1}{2}}})}\left(1+\frac{2\beta_1^2}{1-\beta_1}\right), \tag{70}$$

and

$$\Delta g = g(W_1) - \inf_{W} g(W). \tag{71}$$

In this context, we establish the convergence theory of Algorithms 4 and 5 within the framework of stochastic nonconvex optimization. Chen et al. [31] examined the convergence behavior of the AdaFOM algorithm with a fixed second-order momentum in non-convex optimization scenarios. Their findings suggest an ergodic convergence rate of approximately $\mathcal{O}\left(\frac{\log T + d^2}{\sqrt{T}}\right)$, where T represents the number of iterations and d denoting the dimensionality of the problem.

In a related study, Zhou et al. [32] expanded upon this research by exploring the convergence properties of adaptive gradient methods in non-convex optimization. They specifically investigated the ergodic convergence utilizing AMSGrad with a varying maximum second-order momentum parameter, achieving a convergence rate of $\mathcal{O}(\frac{d^{\frac{1}{2}}}{T^{\frac{1}{2}}} + \frac{d}{T})$

Given the similarity between AMSGrad and PADAM, which is a slight variation thereof, and the emphasis on maintaining a partially adaptive parameter below 0.5, one could adopt Zhou et al.'s approach. By replacing the base learning rate with the proposed linear function $(u.f(H)) + C$, and keeping the partially adaptive parameter fixed at 0.5, an improved convergence rate could be attained. To be more specific, by incorporating a constant value C of our linear function base learning, $\alpha_{base} = u.f(H) + C$ from the convergence rate of Amsgrad in nonconvex $\mathcal{O}(\frac{d^{\frac{1}{2}}}{T^{\frac{1}{2}}} + \frac{d}{T})$, we can achieve an improved convergence rate of $\mathcal{O}(\frac{d^{\frac{1}{2}}}{T^{\frac{1}{2}}} + \frac{d}{T} + \Psi)$ for the proposed ASGD in non-convex setting.

4.2.6 Non-Ergodic Convergence Analysis of the Proposed ASGD

The non-ergodic convergence rate of the proposed ASGD is guarantee by theorem 3 below.

Theorem 4. *The theorem follows the theorem in [3, 104] with the base learning rate replaced with a linear function $uf(H) + C$ and the partially adaptive parameter set at* 0.5.

Suppose assumption 2 and 3 are satisfied, we have the bound

$$\min_{0 \le t \le T} \mathbb{E}\left[||\bigtriangledown g(W_t)||\right] \le \frac{I_1 + I_2\sum_{t=1}^{T-1}(uf(H)+C)_t^2}{\sum_{t=1}^{T}(uf(H)+C)_t^2} \tag{72}$$

where

Algorithm 4 Proposed ASGD (Improved Adam)

```
1:  Input:Base learning rate $\{\alpha_{base}\}$ of the ASGD update, adaptive parameter $\beta_1$, $\beta_2$, preconditioner construction $H$, small constant $0 < \epsilon << 1$ and weight $W$.
2:  Initialize: $W_o$, $H_o$, $n_o$
3:  for $t = 1, ......, T$ do
4:  $z_t = \bigtriangledown g(W_t)$
5:  $n_t = \beta_1 n_{t-1} + (1 - \beta_1) z_t$
6:  $H_t = \beta_2 H_{t-1} + (1 - \beta_2) z_t^2$
7:  $f_{min} = \frac{sgn([|(H_t)|] - [H_t]) + 1}{2}$
8:  $f_{max} = \frac{sgn([H_t] - [|(H_t)|]) + 1}{2}$.
9:  if number of nonzeros$(f_{\max}) < 2*$ number of nonzeros$(f_{\min})$
10: $W_{t+1,small} = \left( W_t - \frac{\alpha_{min}}{\sqrt{(H_t + \epsilon)}} n_t \right)$
11: else:
12: $W_{t+1,large} = \left( W_t - \frac{\alpha_{max}}{\sqrt{(H_t + \epsilon)}} n_t \right)$
13: end
14: end for
15: Return: $W_{t+1}$
```

Algorithm 5 Proposed ASGD (Improved Amsgrad)

```
1:  Input:Base learning rate $\{\alpha_{base}\}$ of the ASGD update, adaptive parameter $\beta_1$, $\beta_2$, preconditioner construction $H$, small constant $0 < \epsilon << 1$ and weight $W$.
2:  Initialize: $W_o$, $H_o$, $\hat{H}_o$, $n_o$
3:  for $t = 1, ......, T$ do
4:  $z_t = \bigtriangledown g(W_t)$
5:  $n_t = \beta_1 n_{t-1} + (1 - \beta_1) z_t$
6:  $H_t = \beta_2 H_{t-1} + (1 - \beta_2) z_t^2$
7:  $\hat{H}_t = \max(\hat{H}_{t-1}, H_t)$
8:  $f_{min} = \frac{sgn\left(\left[|(\hat{H}_t)|\right] - \left[\hat{H}_t\right]\right) + 1}{2}$
9:  $f_{max} = \frac{sgn\left(\left[\hat{H}_t\right] - \left[|(\hat{H}_t)|\right]\right) + 1}{2}$.
10: if number of nonzeros$(f_{\max}) < 2*$ number of nonzeros$(f_{\min})$
11: $W_{t+1,small} = \left( W_t - \frac{\alpha_{min}}{\sqrt{\left(\hat{H}_t + \epsilon\right)}} n_t \right)$
12: else:
13: $W_{t+1,large} = \left( W_t - \frac{\alpha_{max}}{\sqrt{\left(\hat{H}_t + \epsilon\right)}} n_t \right)$
14: end
15: end for
16: Return: $W_{t+1}$
```

$$
\begin{aligned}
& I_1 = [Z^2 + \epsilon]^{\frac{1}{2}} \left(g(W_t) - \min g\right) + [Z^2 + \epsilon]^{\frac{1}{2}} * \\
& \left(\frac{(uf(H) + C)_1 Z^2 \sqrt{d}}{\epsilon^{\frac{1}{2}}(1 - \beta_1)} + \frac{(uf(H) + C)_1 \beta_1 Z^2}{\epsilon^{\frac{1}{2}}(1 - \beta_1)} \right) + \\
& [Z^2 + \epsilon]^{\frac{1}{2}} \left(\frac{(uf(H) + C)_T Z^2}{\epsilon^{\frac{1}{2}}(1 - \beta_1)} \right)
\end{aligned}
\tag{73}
$$

and

$$I_2 = \left[Z^2 + \epsilon\right]^{\frac{1}{2}} \frac{1+\beta_1}{1-\beta_1} \frac{LZ^2}{2\epsilon} \tag{74}$$

If we suppose $\sum_t (uf(H) + C)_t = +\infty$*, then the nonergodic convergence of the gradient sequence can be derived as*

$$\lim_t \bigtriangledown g(W_t) = 0 \quad a.s., \quad \lim_t \mathbb{E}\left[|| \bigtriangledown g(W_t)||^2\right] = 0. \tag{75}$$

In the next sections, we evaluate the performance of our proposed adaptive SGD method against the state-of-the-art methods. We present image classification experiment followed by surgical gesture recognition experiment.

4.3 Image Classification Experiment

We evaluate the strength of our proposed algorithm by comparing it to various baseline methods during the training of the specified network for image classification on CIFAR-100 dataset.

4.3.1 Dataset for Image Classification

The CIFAR-10 [105] dataset is a widely used benchmark in machine learning and computer vision. It consists of 60,000 32x32 color images divided into 10 classes, with each class containing 6,000 images. The dataset is split into 50,000 training images and 10,000 test images. The classes include airplane, automobile (excluding truck or pickup truck), bird, cat, deer, dog, frog, horse, ship, and truck (excluding pickup truck).

The CIFAR-100 dataset [105] features 100 classes, each containing 600 images. For every class, there are 500 training images and 100 testing images. The 100 classes in CIFAR-100 are organized into 20 superclasses. Each image is labeled with both a ”fine” label, indicating its specific class, and a ”coarse” label, denoting the superclass to which it belongs [105].

4.3.2 Network Architecture

We examine two distinct neural network architectures: WideResNet and VGG16-net. WideResNet [13] represents a deep convolutional neural network framework devised to enhance the ResNet (Residual Network) model. It tackles the challenge of training exceedingly deep neural networks by introducing broader layers, capable of capturing more intricate feature representations. Unlike conventional ResNet models that focus on increasing depth, WideResNet prioritizes widening the layers, resulting in a more efficient parameter configuration. The fundamental concept of WideResNet revolves around employing wider convolutional layers with reduced depth, allowing the model to grasp a wider array of features without significantly escalating computational complexity. This strategy aims to mitigate issues such as vanishing gradients and overfitting, commonly encountered in excessively deep networks [13]. On the other hand, VGG16-net [14] is a profound convolutional neural network architecture crafted by the Visual Geometry Group (VGG) at the University of Oxford. It stands out for its simplistic and uniform design, comprising 16 layers, including 13 convolutional layers and three fully connected layers. The convolutional layers utilize compact 3x3 filters with a stride of 1, supplemented by max-pooling layers implemented after every two convolutional layers. VGG16-Net has garnered substantial success in image classification tasks and has served as a foundational model for subsequent advancements in deep learning [14]. At every iteration, the weights of all optimizers are updated, resulting in varying weights for each optimizer across iterations. I did not decouple the weight decay or apply regularization to the weight decay.

4.3.3 Results

Parameter Setting

I perform grid searches to determine optimal parameters for all algorithms, including the base learning rate chosen from $\{0.0001, 0.001, 0.01, 0.1, 1\}$, the partially adaptive parameter (p) from $\{1/4, 1/8, 1/16\}$, and the second-order momentum parameter (β_2) from $\{0.9, 0.99, 0.999\}$. In the case of Adam and Amsgrad, a base learning rate of 0.001 is employed. For Padam, the base learning rate is established as 0.1, and the partially adaptive parameter (p) is chosen as 0.125. We fine-tune the momentum parameters for all adaptive gradient methods. Following the tuning process, the momentum parameters for Adam and Amsgrad are fixed at $\beta_1 = 0.9$ and $\beta_2 = 0.99$. For Padam, the first-order and second-order momentum parameters are set to 0.9 and 0.999. Regarding our proposed ASGD (Improved Amsgrad and Adam), I select the minimum base learning rate as 0.001 and the maximum base learning rate as 0.01. The first-order and second-order momentum parameters for the proposed algorithm are set to 0.9 and 0.999, respectively. In the regard to SUPERADAM [46], I set the parameters as follows: k'=0.1, m'=100, c'=4, γ'=0.04 and τ'=0. I use version 3 for Wada and set the learning rate to 0.001. For both SuperADAM and Wada, the first-order moment value is set to 0.9, respectively. Our assessment of Wada [41] on WideResNet with the CIFAR 100 dataset reveals NAN (Not a Number) values at the initial epoch, indicating divergence due to non-convexity and encountering large gradient problems. Consequently, it was excluded from the results depicted in the Figures

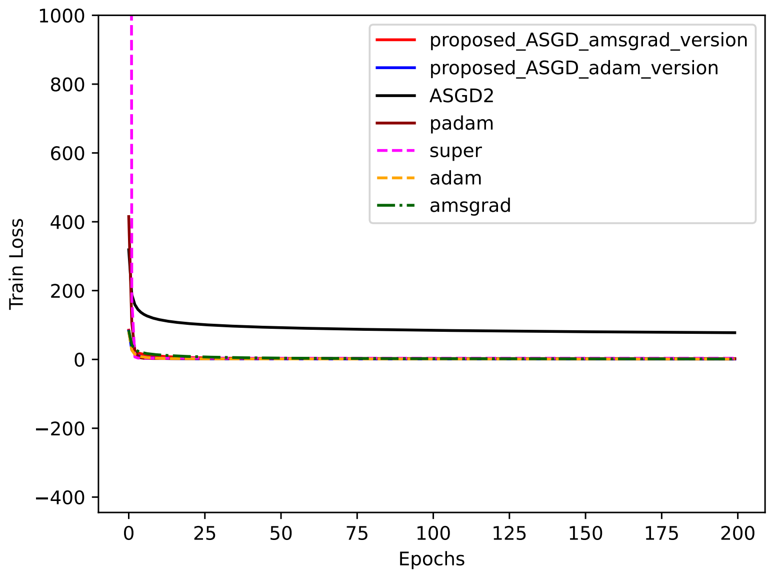


Figure 4.1 Train loss vs. number of epochs with batch size of 32 using WideResNet

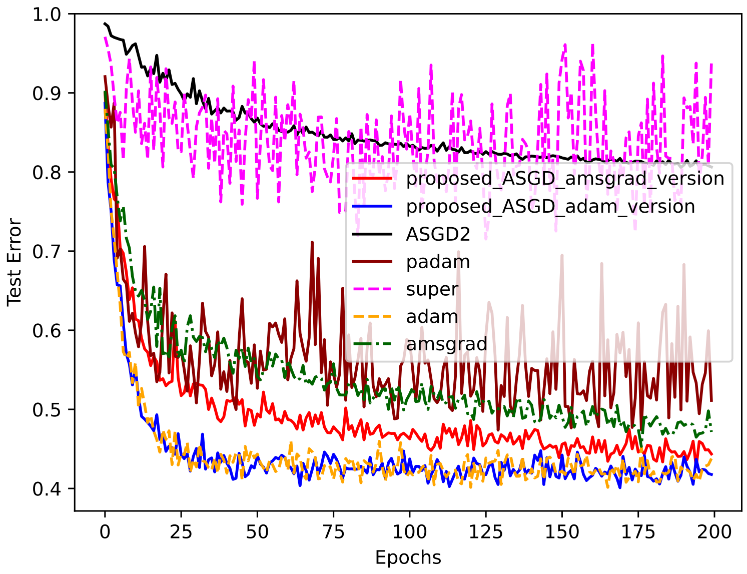
Table 4.2 Final accuracy for the various adaptive stochastic gradient descent algorithms at 200 epochs with a batch size of 32 using WideResNet.

SGD method	ADAM [30]	PADAM [28]	ASGD2 [3]	AMSGRAD [40]	SUPERADAM [46]	Improved Adam	Improved Amsgrad
Accuracy (Top 1)	56.22	48.87	19.37	52.85	6.15	**58.23**	55.64
Accuracy (Top 5)	83.50	79.75	43.49	80.61	20.90	**85.94**	82.69

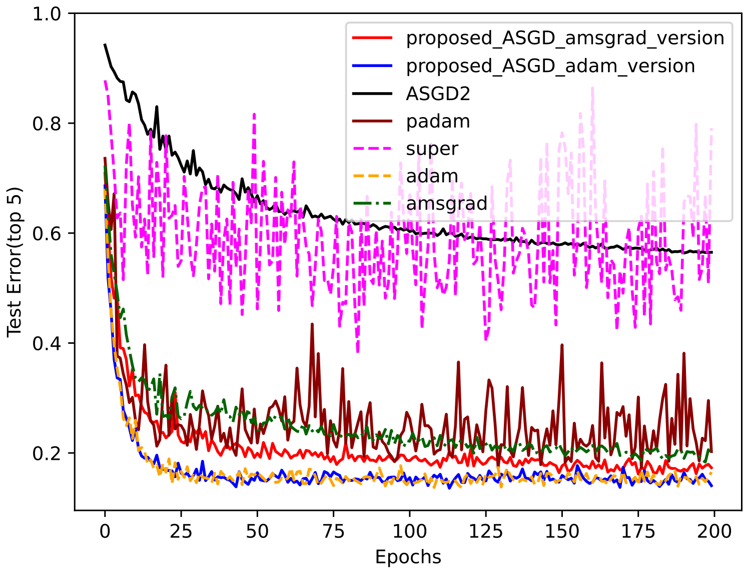
4.1, 4.2a, 4.2b, 4.3, 4.4a, and 4.4b. Also, It should be noted that Wada [41] is tailored for models with rapid gradient decay, making them less effective for shallow models like WideResNet. For ASGD2, the learning rate is set to 0.001, the partially adaptive parameter is fixed at 1×10^{-8}, and the first-order momentum value is set to 0.9.

WideResNet without Data Augmentation

We conducted a comparative analysis, examining the trajectory of train loss and test error (Top 1 and 5) across epochs for various adaptive stochastic gradient descent methods, all applied with a batch size of 32. Figure 4.1 illustrates the progression of train loss over epochs. Following this, Figure 4.2a and 4.2b respectively display the evolution of test error (top 1) and top-5 test error over epochs. In Figure 4.1, it is demonstrated that both the proposed ASGD (Improved Adam) and standard Adam exhibited the lowest training loss among various



(a) Test error (Top-1) vs number of epochs.



(b) Test error (Top-5) vs number of epochs.

Figure 4.2 Accuracy of the various algorithm with batch size of 32 using WideResNet

ASGD methods. Then, we notice that towards the 200th epoch mark, the proposed ASGD (Improved Adam) demonstrated slightly lower top 1 test error than the state-of-the-art ASGD, as shown in Table 4.2. Moreover, the proposed ASGD (Improved Adam) exhibited lower top-5 test error compared to all other methods. Also, Padam's performance degraded in later training stages due to fluctuating coordinate values. Padam combines Adam/Amsgrad and SGD, utilizing Adam/Amsgrad initially and transitioning to SGD later. SGD's is known to be locally instable hence, Padam showed poor testing performance in later stages, evident in both top-1 and top-5 test errors. SUPERADAM exhibited inferior performance compared to all other algorithms because it does not address the small learning rate dilemma problem. Additionally, its poor performance can be attributed to the small batch size used for the evaluation. For WideResNet, we observed that ASGD2 [3] does not effectively address the learning rate dilemma, likely resulting in suboptimal outcomes. Additionally, we had to set the partially adaptive parameter very low to prevent the results from returning NaN values.

Amsgrad maintained stability but suffered from poor generalization later in training due to the maximum second-order momentum selected. The proposed ASGD (Improved Amsgrad) encountered slow convergence alongside Amsgrad due to the large coordinate value size of the maximum second-order momentum matrix. Additionally, AMSGrad's performance declined later in training due to its small base learning rate becoming less effective. Our model (proposed ASGD - Improved Amsgrad) builds upon Amsgrad by categorizing the maximum second-order momentum matrix into large and small components, assigning different base learning rates to stabilize the model in both initial and final training stages, thus ensuring superior generalization performance compared to Amsgrad.

Based on our observations, we find that the maximum second-order momentum matrix leads to coordinate value overshooting in later training stages, particularly notable in WideResNet due to its shallower depth. This phenomenon occurs because the gradient decreases slowly in WideResNet, contributing to poor generalization for both the proposed ASGD (Improved Amsgrad) and standard Amsgrad. Additionally, Adam demonstrates slight oscillations in later training stages (around 150-200 epochs), resulting in poor testing performance attributed to its small base learning rate.

The proposed ASGD (Improved Adam) achieved the best generalization performance for batch sizes 32. By segregating coordinate values into large and small components and assigning different base learning rates, this approach prevents the model from losing learning power in later training stages as shown in Table 4.2.

Figure 4.3 illustrates the progression of train loss across epochs for various adaptive SGD methods with a batch size of 64. Subsequently, the evolution of top-1 test error over epochs is presented, followed by the comparison of top-5 test error over the same period as shown in Figures 4.4a and 4.4b.

Figure 4.3 demonstrates that both the proposed ASGD (Improved Adam) and standard Adam exhibited the

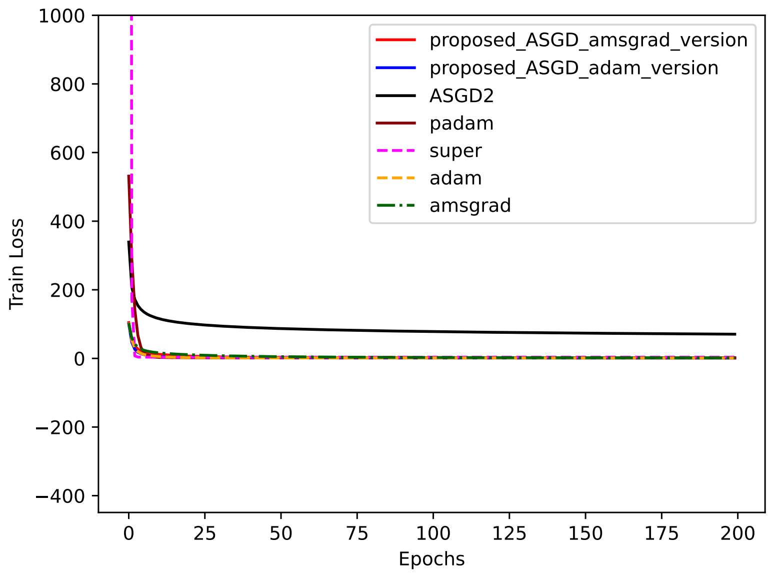
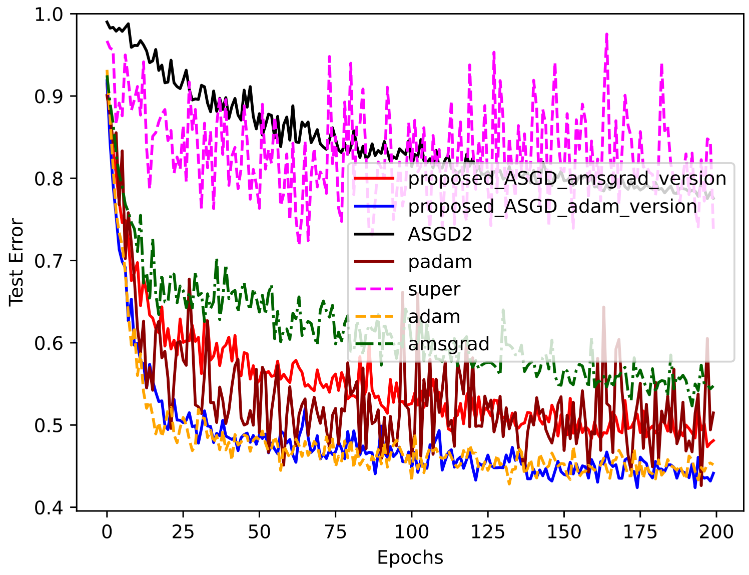


Figure 4.3 Train loss vs. number of epochs with a batch size of 64 using WideResNet

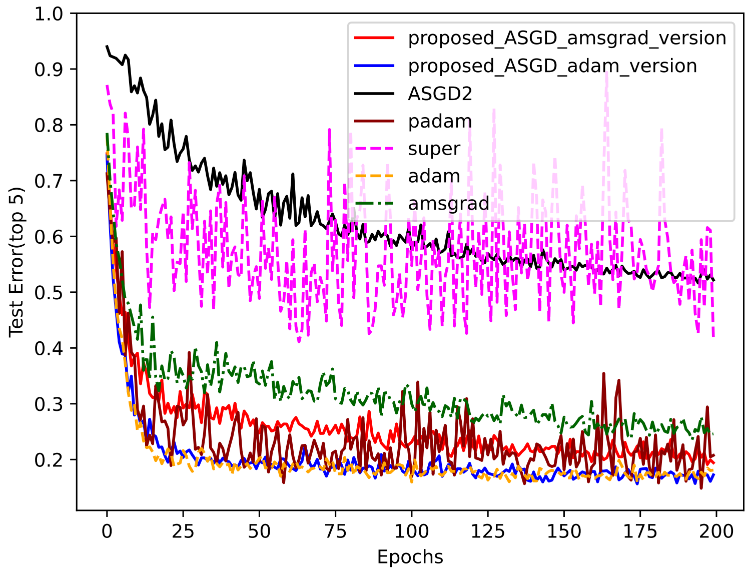
Table 4.3 Final accuracy for the various adaptive stochastic gradient descent algorithms at 200 epochs with a batch size of 64 using WideResNet.

SGD method	ADAM [30]	PADAM [28]	ASGD2 [3]	AMSGRAD [40]	SUPERADAM [46]	Improved Adam	Improved Amsgrad
Accuracy (Top 1)	54.83	48.53	22.44	45.23	26.44	**55.86**	51.89
Accuracy (Top 5)	81.90	79.24	47.82	75.40	58.30	**82.74**	80.58

lowest training loss compared to state-of-the-art ASGD methods. Also, we observe that the convergence rate of Amsgrad slows down when transitioning from a batch size of 32 to 64. Additionally, Adam demonstrates rapid convergence during the initial training stages but exhibits poor testing performance compared to the proposed ASGD(Improved Adam) in later stages. Significantly, all optimizers exhibited diminished accuracy during the later stages of training as the batch size increased from 32 to 64. This phenomenon can be attributed to the trade-off between speed and accuracy that occurs when increasing the batch size. Also, we observed that as we increased the batch size from 32 to 64, Padam performed less than Amsgrad. This was due to Padam's instability at the later stage of training. Moreover, with an increase in batch size, we note an improvement in the performance of SUPERADAM and ASGD2. This enhancement can be attributed to the necessity of variance reduction techniques like SUPERADAM and ASGD2 for larger batch sizes to demonstrate notable improvements, as illustrated in Table 4.3. However, it is notable that SUPERADAM's



(a) Test error (Top-1) vs number of epochs.



(b) Test error (Top-5) vs number of epochs.

Figure 4.4 Test error of the various algorithms with a batch size of 64 using WideResNet

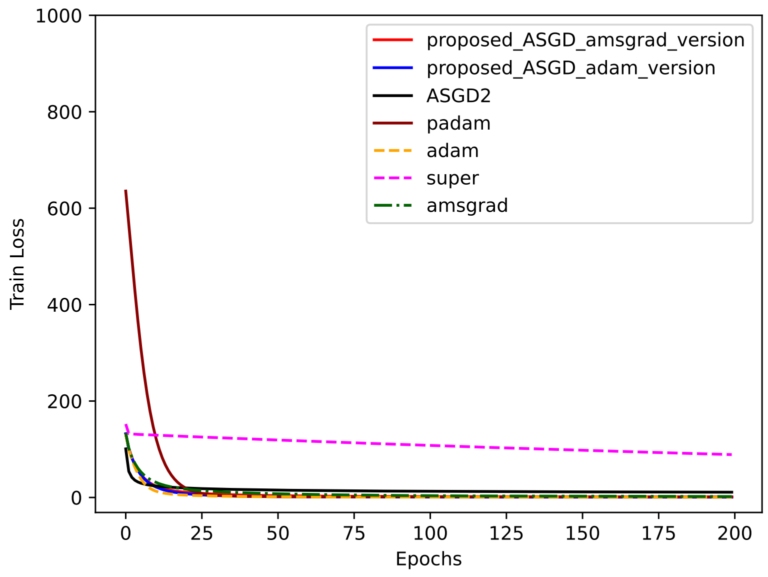


Figure 4.5 Train loss vs. number of epochs with batch size of 256 using WideResNet with data augmentation on CIFAR-100 dataset

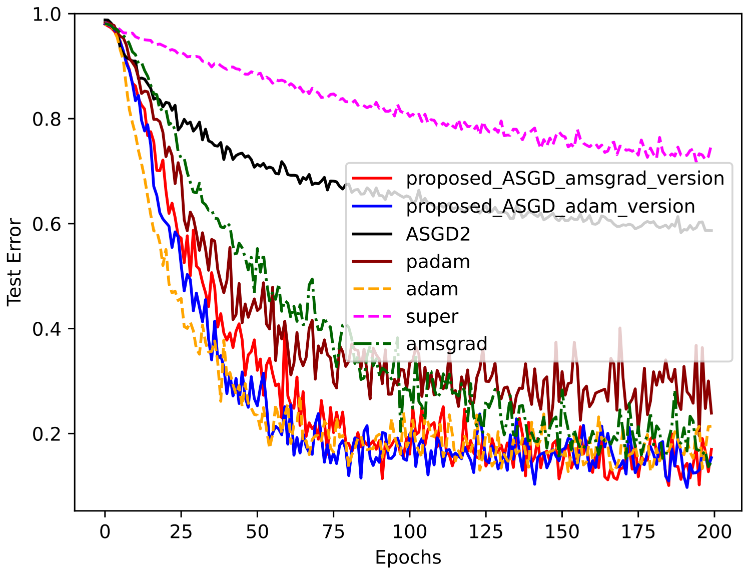
Table 4.4 Final accuracy for the various adaptive stochastic gradient descent algorithms at 200 epochs with a batch size of 256 using WideResNet with data augmentation on CIFAR-100 dataset.

SGD method	ADAM [30]	PADAM [28]	ASGD2 [3]	AMSGRAD [40]	SUPERADAM [46]	Improved Adam	Improved Amsgrad
Accuracy (Top 1)	78.97	76.11	41.13	**85.31**	24.93	84.59	83
Accuracy (Top 5)	97.86	96.04	71.80	**98.84**	52.47	98.04	98.62

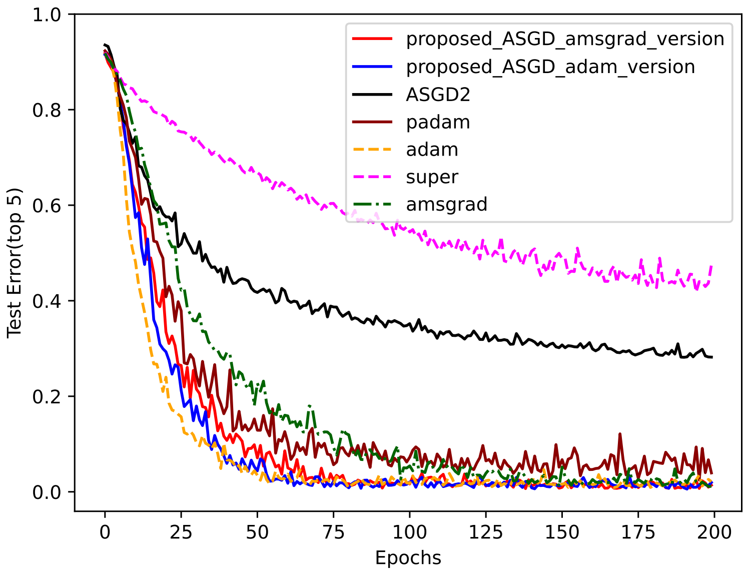
performance fluctuates, indicative of its ineffective resolution of the learning rate dilemma, as depicted in Figures 4.4a and 4.4b. Our proposed ASGD(Improved Adam) achieved the best testing performance of all the methods, as shown in Table 4.3. This demonstrated the resilience of our method across varying batch sizes.

WideResNet with Data Augmentation

A simple mean and standard deviation normalization, sample centering, horizontal flipping, and ZCA whitening to both the training and test datasets was applied. Additionally, a random cropping on the training dataset is performed. The dropout rate is set to 0.3. A batch size of 256 on wideresnet with depth set at 16 and width set at 4 was used. Thee parameter settings is evaluated using the CIFAR-10 and CIFAR-100 datasets.



(a) Test error (Top-1) vs number of epochs.



(b) Test error (Top-5) vs number of epochs.

Figure 4.6 Accuracy of the various algorithm with batch size of 256 using WideResNet with data augmentation on CIFAR-100 dataset

The figures 4.6a, 4.6b and 4.5 illustrate the top-1 test error, top-5 test error, and training loss plotted against the number of epochs using WideResNet with cifar 100. The notable improvement across all models except SuperAdam can be attributed to the data augmentation techniques implemented to raise performance. Additionally, the impact of the optimizers was diminished by the effects of data augmentation and the introduction of dropout. The improved Adam optimizer, used with the WideResNet model and data augmentation, demonstrated greater stability in the later stages of training compared to other models. However, the Amsgrad optimizer achieved the best final top-1 and top-5 accuracy as shown in Table 4.4, albeit with significant fluctuations in model performance. This variability can be linked to Amsgrad's inability to effectively address issues related to the small base learning rate dilemma. SuperAdam struggled because the data augmentation increased the variability of the dataset, making it difficult for the algorithm to effectively reduce variance, which occasionally led to high variance overall. Additionally, Superadam is most effective with rapidly decaying gradient architectures.

The figures 4.8a, 4.8b and 4.7 show the top-1 test error, top-5 test error, and training loss plotted against the number of epochs using cifar 10 with WideResNet.The significant improvement observed across all models except SuperAdam is largely due to the data augmentation techniques applied to enhance performance on the CIFAR-10 dataset as shown in Table 4.5. Furthermore, the influence of the optimizers remained significant, even with the effects of data augmentation and the introduction of dropout in the WideResNet model. The proposed improved Adam optimizer surpassed state-of-the-art optimizers in both stability and testing performance. This achievement can be linked to the strategy of dividing coordinate values and assigning different base learning rates, effectively addressing the small learning rate dilemma. Additionally, optimizers like improved AMSGrad, AMSGrad, and Padam, which utilize the maximum second-order momentum matrix, struggled to perform effectively in WideResNet. This is primarily due to the slow gradient decay, which diminishes the utility of the maximum second-order momentum matrix. Similarly, ASGD2 and SUPERADAM underperformed because they did not adequately tackle the issue of small base learning rates. Furthermore, data augmentation increased the variance of the dataset, resulting in poor performance from SuperAdam. Also, the results show that only ASGD2 overfitted the training dataset, which occurred due to choosing a partial adaptive parameter value either greater than or less than 0.5 as shown in Table 4.5 and 4.6.

VGG16-Net

Figure 4.9 depicts the train loss evolution across epochs for different adaptive SGD methods with a batch size of 64 using the VGG16-Net. Following this, the progression of top-1 test error over epochs is displayed, and the comparison of top-5 test error over the same period is illustrated in Figures 4.10a and 4.10b. The contrast

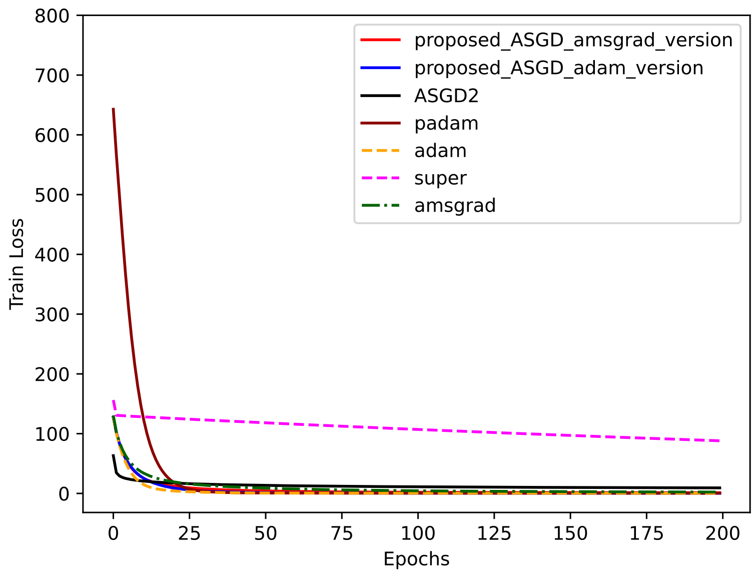
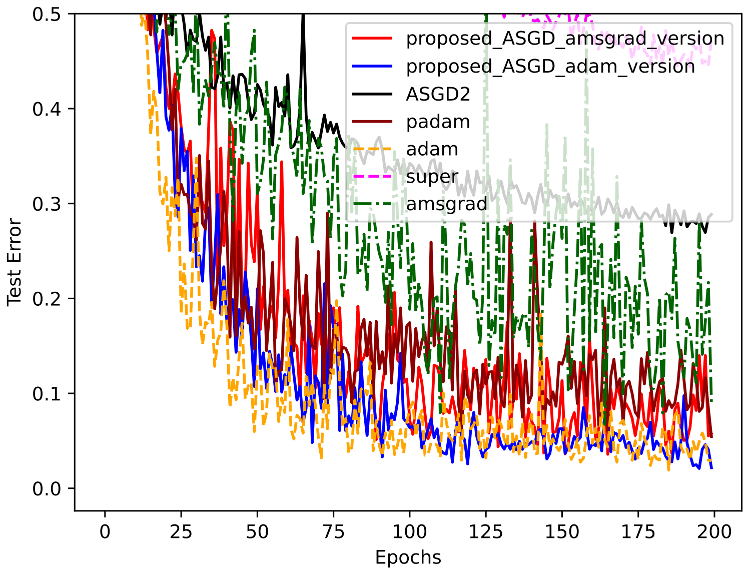


Figure 4.7 Train loss vs. number of epochs with batch size of 256 using WideResNet with data augmentation on CIFAR-10 dataset

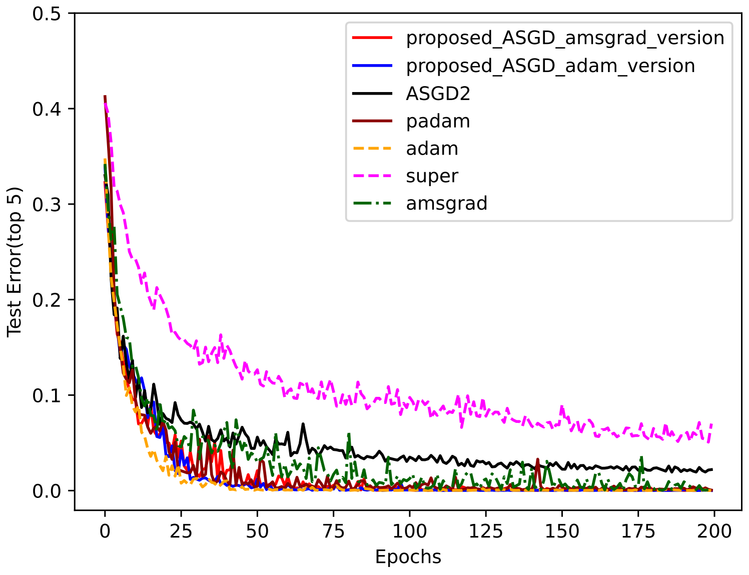
Table 4.5 Final test accuracy for the various adaptive stochastic gradient descent algorithms at 200 epochs with a batch size of 256 using WideResNet with data augmentation on CIFAR-10 dataset.

SGD method	ADAM [30]	PADAM [28]	ASGD2 [3]	AMSGRAD [40]	SUPERADAM [46]	Improved Adam	Improved Amsgrad
Test Accuracy (Top 1)	97.09	94.54	71.14	90.88	52.70	**97.83**	94.23
Test Accuracy (Top 5)	99.98	99.94	97.80	99.80	92.98	**99.99**	99.98

between VGG16-Net and WideResNet lies in their respective depths, with WideResNet being shallower compared to VGG16-Net. Furthermore, WideResNet prioritizes speed over accuracy. During training with VGG16-Net, gradients rapidly diminish, resulting in diminished efficacy for many optimizers due to the small base learning rate, ultimately leading to suboptimal testing performance. While Padam showcases comparable performance with our proposed method (Improved Amsgrad), indications of overfitting (poor generalization) emerge from the gap between training loss and testing accuracy, as shown in Figures 4.9,4.10a and 4.10b. The overfitting arose because, for partially adaptive parameters lower than 0.5, such as in the case of Padam, ASGD performance lacks any identifiable pattern. For this reason, our proposed ASGD algorithm adheres to the partially adaptive parameter of 0.5. Our proposed ASGD (Improved Amsgrad) excels by bolstering generalization through the division of the maximum second-order momentum matrix and



(a) Test error (Top-1) vs number of epochs.



(b) Test error (Top-5) vs number of epochs.

Figure 4.8 Accuracy of the various algorithm with batch size of 256 using WideResNet with data augmentation on CIFAR-10 dataset

Table 4.6 Final Train accuracy for the various adaptive stochastic gradient descent algorithms at 200 epochs with a batch size of 256 using WideResNet with data augmentation on CIFAR-10 dataset.

SGD method	ADAM [30]	PADAM [28]	ASGD2 [3]	AMSGRAD [40]	SUPERADAM [46]	Improved Adam	Improved Amsgrad
Train Accuracy (Top 1)	97.82	96.03	70.65	96.80	54.50	97.78	97.43
Train Accuracy (Top 5)	99.99	99.97	98.03	99.99	94.55	99.99	99.99

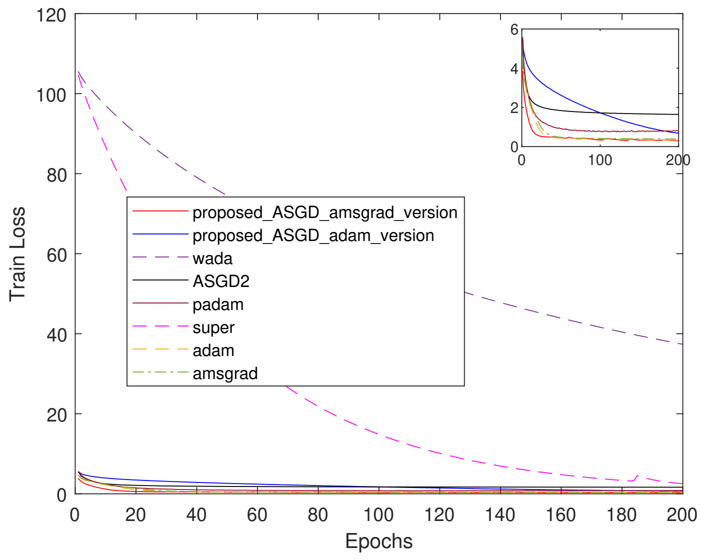
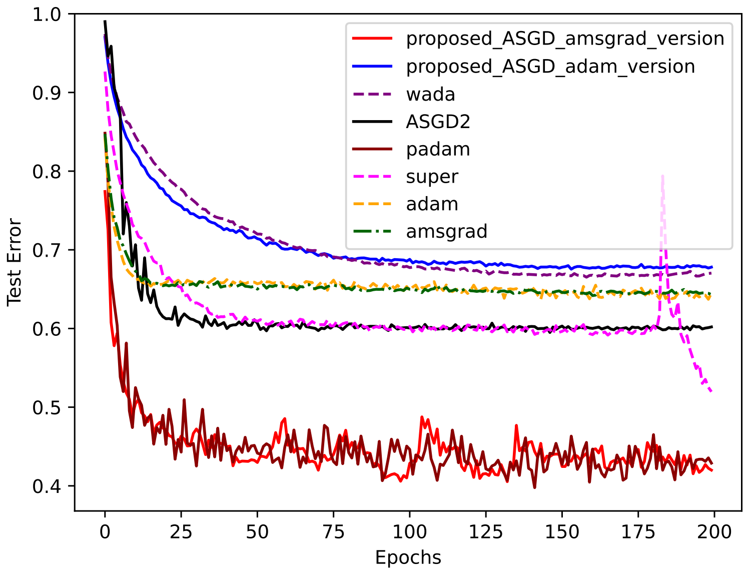


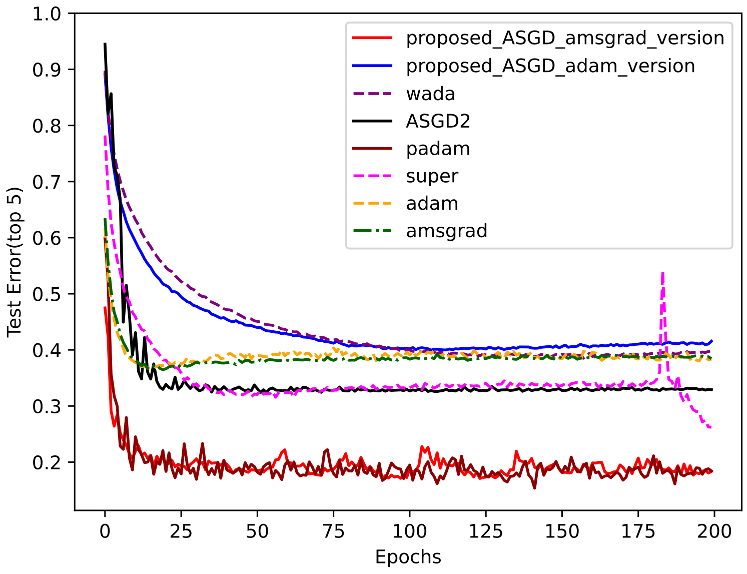
Figure 4.9 Train loss vs. number of epochs with a batch size of 64 using VGG16-Net

allocation of different base learning rates, as shown in Table 4.7. Adam and Amsgrad display poor testing performance attributed to their small base learning rates. Conversely, our proposed ASGD (Improved Adam) fares even worse in testing due to its failure in incorporating the maximum second-order momentum matrix before assigning differing base learning rates. Given the rapid decay in gradients stemming from the depth of the VGG16-Net architecture, attempting to categorize values into small and large groups proves futile within the proposed ASGD (Improved Adam) framework. Also, SuperAdam and Wada encountered challenges due to the small learning rate dilemma. Additionally, SuperAdam exhibited coordinate value overshooting during training, as depicted in Figures 4.9, 4.10a and 4.10b. SUPERADAM also experiences overfitting, indicated by the discernible gap between the training and testing errors. Additionally, the variance reduction technique such as SuperAdam typically necessitates large batches to yield improved results. For VGGNet, ASGD2 exhibited poor generalization when the partially adaptive parameter was greater than 0.5.

In the case of nonconvex functions, optimizing them may yield infinitely many solutions. Therefore, an



(a) Test error (Top-1) vs number of epochs.



(b) Test error (Top-5) vs number of epochs.

Figure 4.10 Test error of the various algorithms with a batch size of 64 using using VGG16-Net

Table 4.7 Final accuracy for the various adaptive stochastic gradient descent algorithms at 200 epochs with a batch size of 32 using VGG16-Net.

SGD method	ADAM [30]	PADAM [28]	ASGD2 [3]	AMSGRAD [40]	SuperAdam [46]	Wada [41]	Improved Adam	Improved Amsgrad
Acccuracy (Top 1)	35.50	57.11	39.80	35.69	48.04	32.96	32.20	**58.00**
Accuracy (Top 5)	61.65	81.58	67.09	61.57	73.77	60.19	58.45	**81.66**

efficient optimization scheme is essential to evade local minima and converge towards a favorable solution. In both WideResNet and VGGNet, we demonstrated that our proposed ASGD method outperforms the current state-of-the-art for nonconvex loss functions in deep learning in terms of convergence rate and generalization performance. This is achieved by categorizing coordinate values into large and small groups and assigning distinct base learning rates, thereby enhancing the learning process. The insights gleaned from this image classification experiment lay the foundation for further research in remote robotic surgery. By leveraging the knowledge acquired, we aim to enhance the reliability of surgical operations conducted remotely, thus advancing the field of robotic surgery. Moreover, in the context of mission critical and tactile Internet applications such as remote robotic surgery demanding high reliability, the convergence of our proposed ASGD to a superior solution is crucial for achieving the objective of delivering precise haptic predictions to the patient domain and accurate feedback predictions to the surgeon domain via the tactile internet connection.

4.4 Remote Surgical Gesture Recognition Experiment

We assess the efficacy of our proposed algorithm by comparing it against various baseline methods during the training of the 2D CNN LSTM surgical gesture recognition model on JIGSAW datasets.

4.4.1 Dataset Description

We use the publicly available JIGSAW dataset in most surgical gesture recognition tasks [4, 25]. The dataset is composed of video and kinematic datasets of three surgical tasks, namely, suturing, knot tying, and needle passing, obtained from the Da Vinci Surgical robot. We focus on suturing tasks with gestures y1, y2, y3, y4, y6, y8, y9, y,10 and y11, as shown in Table 4.1. The suturing dataset was gathered from 8 surgeons with different skill levels; based on Huynhuguyen et al. [53] evaluation, the original dataset was compressed from 480×640 pixels per frame to 240×320 pixel per frame to speed up the training process. The transcripts obtained from the suturing dataset was used to indicate the start and end times of a gesture fed into the recognition model [53]. We compared our proposed optimizer with the state-of-the-art optimizers using a suturing video dataset. To achieve dataset balance, we resample gestures, with particular emphasis on those occurring less frequently, such as y1, y5, y8, y9, and y10 [53]. Assigning 10 frames to each surgeme ensures uniformity, vital given variations in surgeme transcriptions [53]. This standardization is crucial for effective learning, preventing impediments to adaptive SGD optimizer comparisons due to varying lengths. Leveraging expert surgeon data from the suturing video JIGSAW dataset, we split our data into 90% for training and 10% for testing purposes. Then, we introduce a synthetic loss probability of 10% to simulate sparse dataset scenarios and replicate the anticipated poor communication in remote robotic surgery procedures.

4.4.2 Results

Utilizing grid searches, we choose the optimal parameters for all algorithms, encompassing the base learning rate selected from $\{0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1\}$, the partially adaptive parameter (p) from $1/4, 1/8, 1/16$, and the second-order momentum parameter (β_2) from $0.9, 0.99, 0.999$. For Adam and Amsgrad, a base learning rate of 0.0001 is employed. For Padam, the base learning rate is established at 0.001, and the partially adaptive parameter (p) is selected as 0.125. The momentum parameters are fine-tuned for all adaptive gradient methods. Following the tuning process, the momentum parameters for Adam and Amsgrad are set at $\beta_1 = 0.9$ and $\beta_2 = 0.99$. In the case of Padam, the first-order and second-order momentum parameters are assigned values of 0.9 and 0.999. Concerning our proposed ASGD, we designate the minimum base learning rate as 0.0001 and the maximum base learning rate as 0.001. The first-order and second-order momentum parameters for the proposed algorithm are set to 0.9 and 0.999, respectively. In the context of SuperADAM, we establish the parameters as follows: k'=0.1, m'=100, c'=4, γ'=0.04 and τ'=0. Regarding Wada, we employ version 3 with a learning rate set to 0.001. For both SuperADAM and Wada, the first order moment value is configured to 0.9, respectively. For ASGD2 [3], the learning rate is set to 0.001, the partially adaptive parameter is fixed at 1×10^{-8}, and the first-order momentum value is set to 0.9. We employ minibatch gradient descent with batches of size 1.

We employ a model, detailed in Section III.B, designed with ample depth to speed up the gradient descent. In general, class imbalance continues to impact model training even post resampling gestures, as illustrated in Figures 4.11, 4.12a and 4.12b. We notice from Figures 4.11 and 4.12a that, Padam exhibits the poorest generalization, attributed to a disparity between training loss and testing error. Also, the lack of adaptation of the partially adaptive parameter during training may be the reason behind poor generalization performance

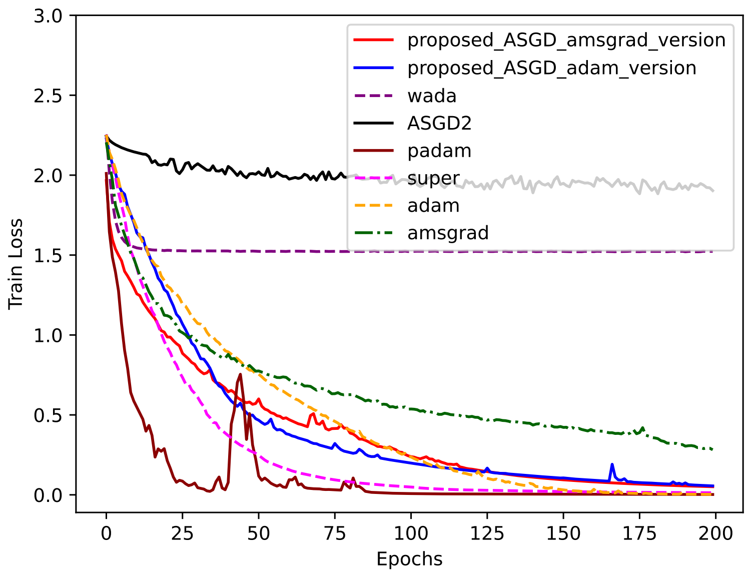
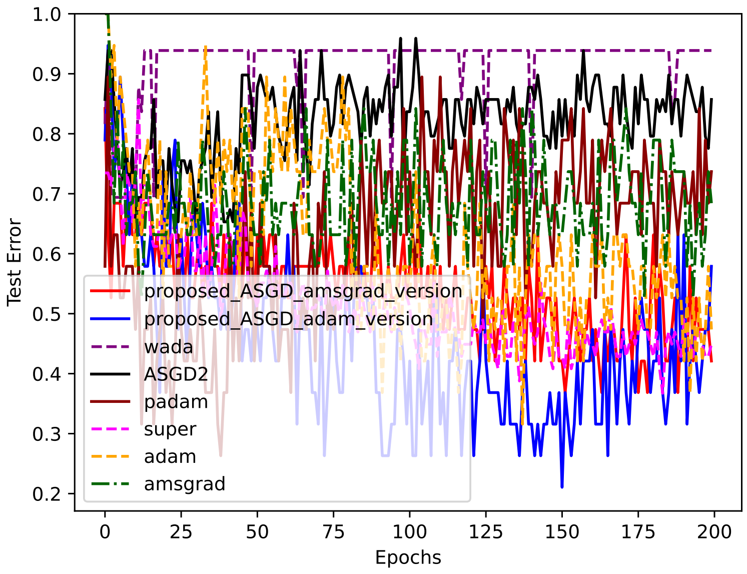


Figure 4.11 Train loss vs. number of epochs with batch size of 1 using 2D CNN LSTM

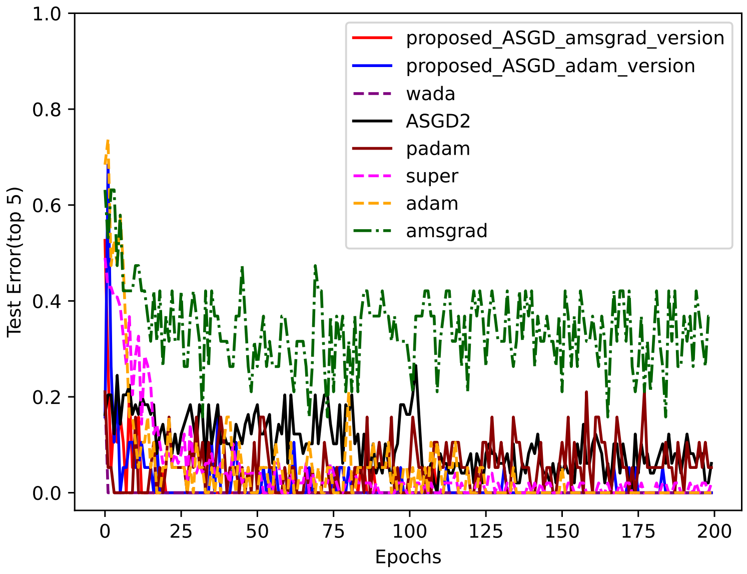
Table 4.8 Final accuracy for the various adaptive stochastic gradient descent algorithms at 200 epochs with a batch size of 1 using 2D CNN LSTM.

SGD method	ADAM [30]	PADAM [28]	ASGD2 [3]	AMSGRAD [40]	SuperAdam [46]	Wada [41]	Improved Adam	Improved Ams-grad
Acccuracy (Top 1)	52.63	26.32	14.29	31.58	55.10	6.12	42.11	**57.89**
Accuracy (Top 5)	**99.999**	94.74	93.88	63.15	97.96	**99.999**	**99.999**	**99.999**

of Padam. Moreover, partially adaptive parameters below 0.5 exhibit performance without any identifiable pattern. While Standard Amsgrad displays greater stability, its testing performance is hindered by a small base learning rate. Amsgrad showed the worst top-5 testing error, as shown in Figure 4.12b and Table 4.8. Adam demonstrates slight but noticeable oscillations in later training stages due to the lack of a maximum second-order momentum matrix. Our proposed ASGD (Improved Amsgrad) yields the highest testing performance. From Figure 4.12a, ASGD (Improved Adam) experiences diminished learning efficacy in later stages due to the failure to integrate the second-order momentum matrix before segregating coordinate values into small and large groups and assigning distinct base learning rates. SuperAdam obtained a lower top-1 accuracy compared to the proposed ASGD (Improved Amsgrad) method as shown in Table 4.8. Also, the drawback of SuperAdam lies in its high instability, stemming from its foundation on Adam, which does not effectively address the challenge of the small learning rate dilemma. In addition, SUPERADAM being a



(a) Test error (Top-1) vs number of epochs.



(b) Test error (Top-5) vs number of epochs.

Figure 4.12 Test error of the various algorithms with a batch size of 1 using using 2D CNN LSTM

variance reduction technique required a large batch size to attain optimal performance. Additionally, Wada's performance suffered due to the same issue of the small learning rate dilemma. On the JIGSAW dataset, ASGD2 demonstrated poor top-1 accuracy due to its inability to handle the small learning rate dilemma and the uncertainty of results when the partially adaptive parameter was set greater than 0.5, as shown in Table 4.8.

4.5 Conclusions

An ASGD approach for training machine learning models for tasks like image classification and surgical gesture recognition is introduced. In the experimentation with image classification using WideResNet and VGG16-Net alongside various optimizers, a significant impact of neural network depth on optimizer design is noted.

For shallow neural networks, the proposed ASGD technique can be executed without the inclusion of the maximum second-order momentum matrix, owing to the inherent slowing of gradient decay. This design choice aims to enhance generalization performance by preventing coordinate overshooting during training after splitting the coordinate values and assigning different base learning rates. Notably, this design ensured that our optimizer outperformed Padam, Amsgrad, and Adam on CIFAR-100 using the WideResNet architecture.

Conversely, in deeper neural networks, gradients tend to diminish more rapidly. In such scenarios, it is advantageous for the optimizer to incorporate the maximum second-order momentum matrix initially before segmenting coordinate values and assigning varied base learning rates. This strategy enhances the optimizer's efficacy in handling deeper networks. Additionally, this design led to our Proposed ASGD outperforming Padam, Amsgrad, and Adam using VGG16-Net architecture.

The surgical gesture recognition application employed a moderately deep model (2D CNN LSTM), and as anticipated, the proposed ASGD (Improved Amsgrad) surpassed state-of-the-art optimizers, showcasing superior performance.

Also, it was observed that SUPERADAM necessitated a large batch size to yield satisfactory results. Additionally, Wada exhibited divergence on non-convex and slowly decaying problems. Moreover, Padam suffered from overfitting due to uncertainty surrounding partially adaptive parameters lower than 0.5. Additionally, ASGD2 demonstrated poor testing performance due to the uncertainty of results when the partially adaptive parameter was set greater than 0.5. Furthermore, both Adam and Amsgrad employed small learning rates to mitigate the learning dilemma problem, resulting in suboptimal results. Our proposed ASGD method achieved the optimal outcome due to our approach to addressing the small learning rate dilemma problem. We accomplished this by partitioning the coordinate values into small and large subsets and assigning different base learning rates, thereby enhancing generalization performance.

In the future, we aim to expand the improvised base learning selection strategy to other models such as AdaBelief Adadelta and RMSprop to assess their performance. Additionally, an intriguing research avenue involves evaluating the performance of the proposed ASGD against a wide variety of neural networks. On top of that, we propose that our ASGD algorithm can be extended and applied across various domains, including 5G/6G networked devices, intelligence cloud, Blockchain, and large-scale networked data centers, to fulfill the requirements of the tactile internet. This extension involves utilizing the Proposed ASGD algorithm to train machine learning models for any tactile internet application, thereby evaluating its performance. Moreover, in the context of tactile internet application, the Proposed ASGD algorithm can be fine-tuned and deployed as an intelligent solution over Mobile Edge Computing (MEC) to address prediction issues.

Convergence Analysis in Convex Setting

To the best of our knowledge, the difference between our proposed algorithm and the existing methods is the introduction of $(uf(H) + C)$ linear function to replace the base learning rate α_{base}. The next steps bear a close resemblance to the steps of proof of Theorem 2 in FastAdamBrief [42] and AMSgrad [40].

Proof: Suppose that $W^* \in R^d$ is classified as the optimal point of the problem. Algorithm 4 and 5 is derived from lemma 2 as follows [42] [40].

$$\begin{aligned} &||H_t^{\frac{1}{4}}(W_{t+1} - W_t^*)||_2^2 \leq ||H^{\frac{1}{4}}(W_t - (uf(H) + C)_t H_t^{-\frac{1}{2}}.n_t - W^*)||_2^2 \\ &= ||H^{\frac{1}{4}}(W_t - (uf(H) + C)_t H_t^{-\frac{1}{2}}.n_t - W^*)||_2^2 - \\ &2(uf(H) + C)\langle \beta_{1t} n_{t-1} + (1 - \beta_{1t}) z_t, W_t - W^* \rangle. \end{aligned} \tag{76}$$

From Theorem 2 and the equation below [40, 42]

$$g_t(W_t) - g_t(W^*) \leq \langle z_t, W_t \rangle \tag{77}$$

we have [40, 42]

$$\begin{aligned} R(T) &= \sum_{t=1}^{T} g_t(W_t) - \min_{W \in \chi} \sum_{t=1}^{T} g_t(W_t) \\ &\leq \sum_{i=1}^{T} \langle z_t, W_t \rangle \end{aligned} \tag{78}$$

Since $H^p(W^*) = W^*$, we can follow AMSgrad [40] and obtain the first inequality and then using Lemma 2 [40, 42], the inequality can be rearranged to obtain [40, 42]

$$\begin{aligned} &\langle z_t, W_t \rangle \leq \\ &\frac{1}{2((u.f(H)+C))_t(1-\beta_{1t})} * \\ &\left[||H^{\frac{1}{4}}(W_{t+1} - W^*)||_2^2 - ||H^{\frac{1}{4}}(W_t - W^*)||_2^2 \right] + \\ &\sum_{t=2}^{T} \frac{\beta_{1t}}{2((u.f(H)+C))_{t-1}(1-\beta_{1t})} ||H^{\frac{-\frac{1}{2}}{2}}(W_t - W^*)||_2^2 + \\ &\sum_{t=2}^{T} \frac{\beta_1((u.f(H)+C))_{t-1}}{2(1-\beta_{1t})} . ||H_{t-1}^{\frac{-1}{4}} n_{t-1}||_2^2 + \\ &\sum_{t=2}^{T} \frac{\beta_1((u.f(H)+C))_t}{2(1-\beta_{1t})} . ||H_t^{\frac{-1}{4}} n_t||_2^2 \end{aligned} \tag{79}$$

and

$$\begin{aligned} R(T) &= R_1 + R_2 + R_3 \\ R_1 &= \sum_{t=1}^{T} \sum_{l=1}^{d} \frac{\beta_{1t}.h_{t,l}^{\frac{1}{4}}}{2((u.f(H)+C))(1-\beta_1)} \left[(W_{t+1} - W^*)^2 - (W_t - W^*)^2 \right], \\ R_2 &= \sum_{t=1}^{T} \sum_{l=1}^{d=1} \frac{\beta_{1t} h_{t,l}^{\frac{1}{2}}}{2((u.f(H)+C))_t(1-\beta_1)} \sum_{t=1}^{T} \sqrt{t} \lambda^{t-1} \\ R_3 &= \frac{1+\beta_1}{2(1-\beta_1)} \sum_{t=1}^{T} \sum_{l=1}^{d} \frac{((u.f(H)+C))_t.n_{t,l}^2}{h_{t,l}^{\frac{1}{2}}}. \end{aligned} \tag{80}$$

According to AMSgrad [40], with the base learning rate α_{base} changed to $((u.f(H)+C))$, the second inequality follows cauchy-schwarz inequality. By applying $\beta_{1t} \leq \beta_1 \leq 1$ which is monotonically decreasing in t and satisfying the condition of convergence, i.e., $\Gamma_t = \sum_{t=1}^{T} \sum_{l=1}^{d} \left(\frac{h_{l,t}^{\frac{1}{2}}}{((u.f(H)+C))_t} - \frac{h_{t-1,l}^{\frac{1}{2}}}{((u.f(H)+C))_{t-1}} \right) \geq$

0, we have

$$R_1 = \frac{1}{2(1-\beta_1)} \sum_{l=1}^{d} \frac{h_{t,l}^{\frac{1}{2}}(w_{1,l} - w_l^*)^2}{((u.f(H)+C))_1} + \\ \sum_{t=1}^{T} \sum_{l=1}^{d} \left(\frac{h_{t,l}^{\frac{1}{2}}}{((u.f(H)+C))_t} - \frac{h_{t-1,l}^{\frac{1}{2}}}{((u.f(H)+C))_{t-1}} \right) (w_{t,l} - w_l), \tag{81}$$

$$\leq \frac{D_\infty^2}{2(1-\beta_1)} \sum_{l=1}^{d} \frac{h_{T,l}^{\frac{1}{2}}}{((u.f(H)+C))_T}, \\ = \frac{D_\infty^2}{2(1-\beta_1)} \sum_{l=1}^{d} \sqrt{T} h_{T,l}^{\frac{1}{2}}. \tag{82}$$

Next, based on AMSgrad [40], for the second term of the equation, we have

$$R_2 = \sum_{t=1}^{T} \sum_{l=1}^{d=1} \frac{\beta_{1t} h_{t,l}^{\frac{1}{2}}}{2((u.f(H)+C))_t(1-\beta_1)} \sum_{t=1}^{T} \sqrt{t}\lambda^{t-1} \leq \\ \frac{\beta_1 d D_\infty^2 Z_\infty^1}{2((u.f(H)+C))(1-\beta_1)(1-\lambda)^2}. \tag{83}$$

Lastly, according to AMSgrad [40], we can utilize geometric series upper bound after relaxing $\sqrt{t}$ to t and we have

$$R_3 = \frac{1+\beta_1}{2(1-\beta_1)} \sum_{t=1}^{T} \sum_{l=1}^{d} \frac{((u.f(H)+C))_t n_{t,l}^2}{h_{t,l}^{\frac{1}{2}}} \leq \\ \frac{(u.f(H)+C) Z_\infty^1 \sqrt{1+\log T}}{(1-\beta_1)^2(1-\gamma)(1-\beta_2)^{\frac{1}{2}}} \sum_{l=1}^{d} ||z_{1:T,l}||_2. \tag{84}$$

□

Ergodic Convergence Analysis in Nonconvex Setting

The subsequent procedures resemble those outlined in the proof of Theorem 3 in [32] and [31], albeit with the base learning rate replaced by a linear function $(uf(H)+C)$. Before delving into the proof of Theorem 3, it is imperative to establish the following lemma

Lemma 3. *[32]: We defined an arbitrary sequence q_t sequence and using the defined n_t we have*

$$
\begin{aligned}
q_t &= W_t - \frac{\beta_1}{1-\beta_1}(W_t - W_{t-1}) \\
&= \frac{1}{1-\beta_1}W_t - \frac{\beta_1}{1-\beta_1}W_t - W_{t-1}.
\end{aligned} \tag{85}
$$

Then, for t greater than 2, we can obtain

$$
\begin{aligned}
&q_{t+1} - q_t = \\
&\frac{\beta_1}{1-\beta_1}\left[I - \left((uf(H)+c)_t H_t^{-\frac{1}{2}}\right)\left((uf(H)+c)_{t-1} H_{t-1}^{-\frac{1}{2}}\right)^{-1}\right] \\
&(W_{t-1} - W_t) - (u.f(H)+C)_t H_t^{-1/2} z_t.
\end{aligned} \tag{86}
$$

Following [32], it should be noted that $q_{t+1} - q_t$ can be represented as n_t, z_t and $\frac{1}{H_t^{\frac{1}{2}}}$. We can reduce $q_{t+1} - q_t$ to

$$
\begin{aligned}
&q_{t+1} - q_t = \\
&\frac{\beta_1}{1-\beta_1}\left((uf(H)+c)_{t-1} H_{t-1}^{-\frac{1}{2}} - (uf(H)+c)_t H_t^{-\frac{1}{2}}\right) \\
&n_t - (u.f(H)+C)_t H_t^{-1/2} z_t.
\end{aligned} \tag{87}
$$

Proof of Theorem 3(slight modification of [32])

Based on the steps in [32], considering L is smooth, we have

$$
g(q_{t+1}) \leq g(q_t) + R_4 + R_5 + R_6, \tag{88}
$$

where

$$
\begin{aligned}
R_4 =& \frac{\beta_1}{1+\beta_1} \triangledown g(W_t)^T \left((u.f(H)+C)_{t-1} H_{t-1}^{-\frac{1}{2}}\right) n_{t-1} \\
&- \triangledown g(W_t)^T (u.f(H)+C)_t H_t^{-\frac{1}{2}} z_t.
\end{aligned} \tag{89}
$$

Given that

$$
\begin{aligned}
&\bigtriangledown g(W_t)^T \left((u.f(H)+C)_{t-1} H_{t-1}^{-\frac{1}{2}} - (u.f(H)+C)_t H_t^{\frac{-1}{2}} z_t \right) n_{t-1} H_{t-1}^{-\frac{1}{2}} \\
&\leq || \bigtriangledown g(W_t)||_\infty . ||(uf(H)+c)_{t-1} H_{t-1}^{-\frac{1}{2}} - (uf(H)+c)_t H_t^{-\frac{1}{2}} || \\
&||n_{t-1}||_\infty , \\
&\leq Z_\infty^2 \left[||(uf(H)+c)_{t-1} H_{t-1}^{-\frac{1}{2}} ||_{1,1} - ||(uf(H)+c)_t H_t^{-\frac{1}{2}} ||_{1,1} \right] .
\end{aligned}
\tag{90}
$$

According to generalized Holder inequality expression in [32], for $x, y, z \in \Re^d$, we can obtain

$$
x^T A y \leq ||x||_\infty * ||A||_{1,1} * ||y||_\infty , \tag{91}
$$

then the the bound is found to be

$$
\begin{aligned}
&- \bigtriangledown g(W_t)^T (u.f(H)+C)_t H_t^{-\frac{1}{2}} z_t \\
&= - \bigtriangledown g(W_t)^T (u.f(H)+C)_{t-1} z_t \\
&- g(W_t)^T (u.f(H)+C)_t H_t^{-\frac{1}{2}} - g(W_t)^T (u.f(H)+C)_{t-1} H_{t-1}^{-\frac{1}{2}} , \\
&\leq -g(W_t)^T (u.f(H)+C)_{t-1} H_{t-1}^{-\frac{1}{2}} z_t + \\
&Z_\infty^2 \left[||(uf(H)+c)_{t-1} H_{t-1}^{-\frac{1}{2}} ||_{1,1} - ||(uf(H)+c)_t H_t^{-\frac{1}{2}} ||_{1,1} \right] .
\end{aligned}
\tag{92}
$$

Following the steps in [31], bound for R_5 can be established as

$$
\begin{aligned}
&R_5 = \left(\bigtriangledown g(q_t) - \bigtriangledown g(W_t) \right)^T (q_{t+1} - q_t) \\
&= L \frac{\beta_1}{1-\beta_1} ||(uf(H)+c)_t H_t^{-\frac{1}{2}} z_t ||_2 ||W_t - W_{t-1}||_2 + \\
&L \left(\frac{\beta_1}{1-\beta_1} \right) ||W_t - W_{t-1}||_2^2 , \\
&\leq L ||(uf(H)+c)_t H_t^{-\frac{1}{2}} z_t ||_2^2 + 2L \left(\frac{\beta_1}{1-\beta_1} \right)^2 ||W_t - W_{t-1}||_2^2 .
\end{aligned}
\tag{93}
$$

According to [31], bound for R_6 can be computed as

$$\begin{aligned}
R_6 &= \frac{L}{2}||q_{t+1} - q_t||_2^2 \\
&\leq \frac{L}{2}\left[||uf(H)+c)_t H_t^{-\frac{1}{2}} z_t||_2 + \frac{\beta_1}{1-\beta_1}||W_{t-1} - W_t||_2\right]^2, \\
&\leq L||uf(H)+c)_t H_t^{-\frac{1}{2}} z_t||_2^2 + 2L\left(\frac{\beta_1}{1-\beta_1}\right)^2 ||W_{t-1} - W_t||_2^2.
\end{aligned} \tag{94}$$

Based on [31], bound for $g(q_t+1) - g(q_t)$ can be expressed as

$$\begin{aligned}
&g(q_t+1) - g(q_t) \leq -\bigtriangledown g(W_t)^T (uf(H)+c)_t H_t^{-\frac{1}{2}} z_t + \\
&2L||(uf(H)+c)_t H_t^{-\frac{1}{2}} n_t||_2^2 \\
&+ 4L\left(\frac{\beta_1}{1-\beta_1}\right)^2 ||(uf(H)+c)_{t-1} H_{t-1}^{-\frac{1}{2}} n_{t-1}||_2^2.
\end{aligned} \tag{95}$$

For $t = 2$ to T

$$\begin{aligned}
&Z_\infty^{-1} \sum_{t=2}^{T} (uf(H)+c)_{t-1} \,\mathbb{E}\,||\bigtriangledown g(W_t)||_2^2 \\
&\leq \mathbb{E}\left[g(q_1) + \frac{Z_\infty^2 ||(u.f(H)+C)_1 H_1^{-\frac{1}{2}}||_{1,1}}{1-\beta_1}\right] - \\
&\left[g(q_{T+1}) - \frac{Z_\infty^2 ||(u.f(H)+C)_T H_T^{-\frac{1}{2}}||_{1,1}}{1-\beta_1}\right] + \\
&2L\sum_{t=1}^{T} \mathbb{E}\,||(uf(H)+c)_t H_t^{-\frac{1}{2}} z_t||_2^2 \\
&+4L\sum_{t=2}^{T} \mathbb{E}\left(\frac{\beta_1}{1-\beta_1}\right)^2 ||(uf(H)+c)_{t-1} H_{t-1}^{-\frac{1}{2}} n_{t-1}||_2^2 \\
&\leq \mathbb{E}\left[\bigtriangledown g + \frac{Z_\infty^2 (u.f(H)+C)_1 \epsilon^{-\frac{1}{2}} d}{1-\beta_1} + d(uf(H)+C)_1 Z_\infty\right] + \\
&2L\sum_{t=1}^{T} \mathbb{E}\,||(uf(H)+c)_t H_t^{-\frac{1}{2}} z_t||_2^2 \\
&+4L\sum_{t=2}^{T} \mathbb{E}\left(\frac{\beta_1}{1-\beta_1}\right)^2 ||(uf(H)+c)_{t-1} H_{t-1}^{-\frac{1}{2}} n_{t-1}||_2^2.
\end{aligned} \tag{96}$$

Following [32], with $\gamma = \frac{\beta_1}{\beta_2^{\frac{1}{2}}}$, we can obtain

$$\begin{aligned}&\sum_{t=1}^{T}(u.f(H)+C)_t^2\,\mathbb{E}\left[||H_t^{-\frac{1}{2}}n_t||_2^2\right]\\&\leq\frac{T^{\frac{1}{2}}(uf(H)+C)_t^2(1-\beta_1)}{2\epsilon^{\frac{1}{2}}(1-\beta_2)^{\frac{1}{2}}(1-\gamma)}\,\mathbb{E}\left(\sum_{i=1}^{d}||z_{1:T,i}||_2\right)\end{aligned}\tag{97}$$

and

$$\begin{aligned}&\sum_{t=1}^{T}(u.f(H)+C)_t^2\,\mathbb{E}\left[||H_t^{-\frac{1}{2}}z_t||_2^2\right]\\&\leq\frac{T^{\frac{1}{2}}(uf(H)+C)_t^2}{2\epsilon^{\frac{1}{2}}(1-\beta_2)^{\frac{1}{2}}(1-\gamma)}\,\mathbb{E}\left(\sum_{i=1}^{d}||z_{1:T,i}||_2\right).\end{aligned}\tag{98}$$

According to [31], given $\kappa\in[\max\{0,4p-1\},1]$ and after introducing equation (97) and (98) into 88, we have

$$\begin{aligned}&\mathbb{E}||\bigtriangledown g(W_{out})||_2^2=\frac{1}{\sum_{t=2}^{T}(uf(H)+C)_{t-1}}\\&\sum_{t=2}^{T}(uf(H)+C)_{t-1}\,\mathbb{E}||\bigtriangledown g(W_t)||_2^2\\&\leq\frac{Z_\infty}{\sum_{t=2}^{T}(uf(H)+C)_{t-1}}*\\&\left[\bigtriangledown g+\frac{Z_\infty^2(u.f(H)+C)_1\epsilon^{-\frac{1}{2}}d}{1-\beta_1}+d(uf(H)+C)_1Z_\infty\right]+\\&\sum_{t=1}^{T}\frac{2LZ_\infty}{(uf(H)+C)_{t-1}}\frac{T^{\frac{1}{2}}(uf(H)+C)_t^2}{2\epsilon^{\frac{1}{2}}(1-\beta_2)^{\frac{1}{2}}(1-\gamma)}\,\mathbb{E}\left(\sum_{i=1}^{d}||z_{1:T,i}||_2\right)^{1-\kappa}+\\&\sum_{t=2}^{T}\frac{4LZ_\infty}{(uf(H)+C)_{t-1}}\left(\frac{\beta_1}{1-\beta_1}\right)^2*\\&\frac{T^{\frac{1}{2}}(uf(H)+C)_t^2(1-\beta_1)}{2\epsilon^{\frac{1}{2}}(1-\beta_2)^{\frac{1}{2}}(1-\gamma)}\,\mathbb{E}\left(\sum_{i=1}^{d}||z_{1:T,i}||_2\right)^{1-\kappa}\\&\leq\frac{1}{T(uf(H)+C)}2Z_\infty\bigtriangledown g+\frac{2}{T}\left(\frac{Z_\infty^3\epsilon^{-\frac{1}{2}}d}{1-\beta_1}+dZ_\infty^2\right)+\\&\left(\frac{2Z_\infty L(uf(H)+C)}{T^{\frac{1}{2}}\epsilon^{\frac{1}{2}}(1-\gamma)(1-\beta_2)^{\frac{1}{2}}}\right)\mathbb{E}\left(\sum_{i=1}^{d}|||z_{1:T,i}||_2\right)\left(1+2(1-\beta_1)\left(\frac{\beta_1}{1-\beta_1}\right)^2\right).\end{aligned}\tag{99}$$

According to [32], since $\alpha_t = uf(H) + C$, we can state theorem 3 with condition $||z_{1:T,i}||_2 \leq Z_\infty T^s$ as

$$\frac{1}{T-1}\sum_{t=2}^{T}\mathbb{E}[|| \bigtriangledown g(W_t)||_2^2] \leq \frac{R_8}{T.(u.f(H)+C)} + \frac{R_9 d}{T} + \frac{(u.f(H)+C)R_{10}d}{T^{\frac{1}{2}}}, \tag{100}$$

where

$$R_8 = 2Z_\infty \Delta g, \tag{101}$$

$$R_9 = \frac{2Z_\infty^3 \epsilon^{-\frac{1}{2}}}{1-\beta_1} + 2Z_\infty^2, \tag{102}$$

$$R_{10} = \frac{2LZ_\infty^2}{\epsilon^{\frac{1}{2}}(1-\beta_2)^{\frac{1}{2}}(1-\frac{\beta_1}{\beta_2^{\frac{1}{2}}})}\left(1+\frac{2\beta_1^2}{1-\beta_1}\right), \tag{103}$$

and

$$\Delta g = g(W_1) - \inf_W g(W). \tag{104}$$

□

Non-Ergodic Convergence Analysis

Following the work in [3, 104] with base learning rate modified to a linear function $uf(H) + C$ and the partially adaptive parameter fixed at 0.5, we begin the proof of Theorem 4 by stating slightly modified lemma taken from [3, 104].

Lemma 4. *: Assuming $(W_t)_{t\leq 1}$ is generated by algorithm 4 and 5 as*

$$\theta_t = \mathbb{E}\left(\langle -(uf(H)+C)z(W_t), \frac{\beta_1 n_{t-1} + (1-\beta_1)z_t}{(H_t+\epsilon)^{\frac{1}{2}}}\rangle\right), \tag{105}$$

then

$$\theta_t \leq \beta_1\theta_{t-1} - (uf(H)+C)_t\frac{(1-\beta_1)}{(Z^2+\epsilon)^{\frac{1}{2}}}\,\mathbb{E}\,||z(W_t)||^2 + R_{7t}, \tag{106}$$

where

$$\begin{aligned} R_{7t} = &((uf(H)+C)_{t-1} - (uf(H)+C)_t)\frac{\beta_1 Z^2}{\epsilon^{\frac{1}{2}}} + \\ &\frac{\beta_1 L Z^2}{\epsilon}(uf(H)+C)_t^2 + \\ &(uf(H)+C)_t Z^2 \sqrt{d}\,\mathbb{E}\left[\sum_{j=1}^{d}\left(\frac{1}{(h_{j,t-1}+\epsilon)} - \frac{1}{(h_{j,t}+\epsilon)}\right)\right]. \end{aligned} \tag{107}$$

Then, we have

$$\begin{aligned} &(1-\beta_1)\sum_{t=1}^{T}(u.f(H)+C)_t\frac{\mathbb{E}\,||z(W_t)||^2}{(Z^2+\epsilon)^{\frac{1}{2}}} \\ &\leq -\theta_t + (1-\beta_1)\sum_{t=1}^{T-1}(-\theta_t) + \sum_{t=1}^{T}(R_{7t}) \\ &\leq (1-\beta_1)\sum_{t=1}^{T-1}(-\theta_t) + \sum_{t=1}^{T}(R_{7t}) + \frac{(uf(H)+C)_T Z^2}{\epsilon^{\frac{1}{2}}}. \end{aligned} \tag{108}$$

Based on the steps in [3, 104], considering the gradient z is Lipschitz continuous at point W_t, we have

$$\begin{aligned} &g(W_{t+1}) \leq g(W_t) + \langle z(W_t), W_{t+1} - W_t\rangle + \frac{L}{2}||W_{t+1} - W_t||^2, \\ &= g(W_t) - (uf(H)+C)_t\langle z(W_t), \frac{n_t}{(H_t+\epsilon)^{\frac{1}{2}}}\rangle \\ &+ \frac{L(uf(H)+C)_t^2}{2}||\frac{n_t}{(H_t+\epsilon)^{\frac{1}{2}}}||^2, \end{aligned} \tag{109}$$

and the total expectation is computed as

$$g(W_{t+1}) \leq g(W_t) + \theta_t + \frac{LZ^2(uf(H)+C)^2}{2\epsilon}. \tag{110}$$

Given that

$$\begin{aligned} &(1-\beta_1)\sum_{t=1}^{T}(uf(H)+C)\frac{\mathbb{E}\,||\bigtriangledown g(W_t)||^2}{(Z^2+\epsilon)^{\frac{1}{2}}} \\ &\leq (1-\beta_1)(g(W_1) - \min g) + \frac{LZ^2}{2\epsilon}(1-\beta_1)\sum_{t=1}^{T-1}(uf(H)+C)_t + \\ &\sum_{t=1}^{T} R_{7t} + (uf(H)+C)_T\frac{Z^2}{\epsilon^{\frac{1}{2}}}, \end{aligned} \tag{111}$$

and

$$\sum_{t=1}^{T} R_{7t} \leq \frac{\beta_1 L Z^2}{\epsilon} \sum_{t=1}^{T} (uf(H)+C)_t^2 + \\ \frac{(uf(H)+C)_1 \beta_1 Z^2}{\epsilon^{\frac{1}{2}}} + \frac{(uf(H)+C)_1 Z^2 \sqrt{d}}{\epsilon^{\frac{1}{2}}}, \tag{112}$$

we arrive at

$$\sum_{t=1}^{T-1} (uf(H)+C)_t^2 \, \mathbb{E} \, ||z(W_t)||^2 \leq \\ [Z^2+\epsilon]^{\frac{1}{2}} (g(W_1) - \min g) + [Z^2+\epsilon]^{\frac{1}{2}} * \\ \left(\frac{(uf(H)+C)_1 Z^2 \sqrt{d}}{\epsilon^{\frac{1}{2}} (1-\beta_1)} + \frac{(uf(H)+C)_1 \beta_1 Z^2}{\epsilon^{\frac{1}{2}} (1-\beta_1)} \right) + \\ [Z^2+\epsilon]^{\frac{1}{2}} \left(\frac{(uf(H)+C)_T Z^2}{\epsilon^{\frac{1}{2}} (1-\beta_1)} \right) + \\ \left[Z^2+\epsilon\right]^{\frac{1}{2}} \frac{1+\beta_1}{1-\beta_1} \frac{L Z^2}{2\epsilon} \sum_{t=1}^{T-1} (uf(H)+C)_t^2. \tag{113}$$

□

Chapter 5

Conclusion, Limitations and Future Work

5.1 Conclusion

Tactile Internet applications, such as remote robotic surgery, hold the potential to enhance human life significantly. However, their implementation is still in its infancy primarily due to demanding performance prerequisites, notably ultra-low latency and ultra-high reliability. As a result, novel machine learning models have been devised to bolster reliability, ensure safety, and reduce latency in remote robotic surgery. This thesis is dedicated to improve safety, reliability and reduce latency in remote robotic surgery. To tackle this challenges, cutting-edge machine learning techniques designed to efficiently handle delayed or lost data during real-time surgical procedures is introduced. The algorithmic prerequisites necessary for creating responsive and predictive solutions in the field of remote robotic surgery, analyzing the latest advancements in the domain are discussed.

Through our research endeavors, groundbreaking machine learning framework leveraging innovative low-rank matrix factorization methods to reactively address delayed or lost packets in remote robotic surgery are unveiled. Subsequently, extensive simulations to evaluate the performance of these algorithms are conducted. Additionally, the capability of our proposed frameworks to fulfill the stringent requirements of remote robotic surgery procedures are assessed. Moreover, the low-rank matrix factorization method with a convex formulation results in a suboptimal solution as it approximates the original nonconvex problem are noted. Additionally, the nonconvex formulation of the low-rank matrix factorization necessitates a favorable initial point to achieve optimal results. The proposed approach achieves the best results because we utilize a convex formulation of the low-rank matrix factorization technique as initialization for the nonconvex formulation of the low-rank matrix factorization method.

Then, a novel adaptive stochastic gradient descent algorithm for training machine learning models in remote robotic surgery using video dataset is proposed. Furthermore, theoretical guarantees of the proposed ASGD in convex and nonconvex settings are analyzed. Furthermore, the studies have showcased that our ASGD approach surpasses the current state-of-the-art in deep learning concerning convergence rate and generalization performance, particularly for nonconvex loss functions. The extensive evaluations have demonstrated that our proposed method surpasses the performance of state-of-the-art optimizers in surgical gesture recognition experiment. We observed that SUPERADAM necessitated a large batch size to yield satisfactory results. Additionally, Wada exhibited divergence on non-convex and slowly decaying problems. Moreover, Padam suffered from overfitting due to uncertainty surrounding partially adaptive parameters lower than 0.5. Additionally, ASGD2 demonstrated poor testing performance due to the uncertainty of results when the partially adaptive parameter is set greater than 0.5. Furthermore, both Adam and Amsgrad employed small learning rates to mitigate the learning dilemma problem, resulting in suboptimal results. The proposed ASGD method achieved the optimal outcome due to our approach to addressing the small learning rate dilemma problem. It is accomplished by partitioning the coordinate values into small and large subsets and assigning different base learning rates, thereby enhancing generalization performance. In addition, the CIFAR-10 and CIFAR-100 datasets are augmented and compared with various optimizers using the WideResNet architecture. The evaluation demonstrated that the proposed improved Adam optimizer yielded more stable results compared to the state-of-the-art optimizers.

5.2 Limitations of the Proposed Algorithms

In the evaluation of the proposed algorithm, it has been demonstrated that the algorithms proposed in this thesis offer numerous potential advantages for remote robotic surgery in the context of 5G. Nevertheless, certain limitations persist. This subsection delineates those limitations of the proposed algorithms. In our initial contribution, a predetermined number of iterations was employed to alternate between convex relaxed low-rank matrix factorization and nonconvex low-rank matrix factorization for scaling machine learning. Consequently, the model might not adequately respond to the real-time sensitivity of remote robotic surgery, failing to adapt optimally to data dynamics or surgical contexts.

Regarding our subsequent contribution limitations, our algorithm lacks the capability to automatically activate the maximum second-order momentum matrix when utilizing a deep learning model and deactivate the maximum second-order momentum matrix when employing a shallow learning model to improve the performance of remote robotic surgery. Moreover, in the subsequent contribution, our proposed ASGD does not adequately address class imbalance issue associated with the JIGSAW robotic surgery dataset. Furthermore, with the subsequent contribution, the proposed ASGD does not perform effectively when dealing with dense gradients. On top of that, the limited availability of publicly accessible robotic surgery data for training the model has impeded our ability to pinpoint limitations in the algorithm for future enhancement.

The proposed algorithm may encounter challenges in meeting stringent latency requirements, particularly as they become more demanding. This limitation stems from the absence of additional edge resources, such as a distributed edge or federated learning framework, which could accelerate the processing time of the machine learning model for remote robotic surgery. Moreover, the algorithm may face difficulties in meeting reliability requirements, especially as they become more stringent. This limitation is attributed to the need for enhanced edge resources to ensure accurate data collection within the proposed framework. As a result, the algorithm lacks an efficient mechanism to satisfy the reliability requirement under stringent constraints.

5.3 Future Work

This thesis provides insights about the future research directions required to meet the requirements of Tactile Internet. The work discussed here opens up further avenues for research. For the first contribution, we can combine sparse Gaussian Process Regression with nonconvex regularization and more computationally-efficient convex preserving strategies to further improve their predictive performance. In addition, edge servers can used to decrease the update time of the proposed framework. Finally, an interesting future research avenue would be to evaluate different machine learning methods for a broader range of surgical tasks based on different experimental setups in a 5G-enabled Tactile Internet setting.

For the second contribution, our objective is to extend the improvised base learning selection strategy to encompass additional models like FastAdaBelief. This extension entails leveraging strong convexity to attain accelerated convergence rates. Additionally, an intriguing research avenue involves evaluating the performance of the proposed ASGD against a wide variety of neural networks. On top of that, we propose that our ASGD algorithm can be extended and applied across various domains, including 5G/6G networked devices, intelligence cloud, blockchain, and large-scale networked data centers, to fulfill the requirements of the tactile internet. This extension involves utilizing the proposed ASGD algorithm to train machine learning models for any tactile internet application, thereby evaluating its performance. Moreover, in the context of tactile internet application, the proposed ASGD algorithm can be fine-tuned and deployed as an intelligent solution over Mobile Edge Computing to address latency issues.

Bibliography

[1] T. Zhang and Y. Yang, “Robust PCA by Manifold Optimization,” *Journal of Machine Learning Research*, vol. 19, no. 80, pp. 1–39, 2018. [Online]. Available: http://jmlr.org/papers/v19/17-473.html

[2] X. Yi, D. Park, Y. Chen, and C. Caramanis, “Fast algorithms for robust PCA via gradient descent,” *Advances in neural information processing systems*, vol. 29, 2016.

[3] T. Sun, L. Qiao, Q. Liao, and D. Li, “Novel Convergence Results of Adaptive Stochastic Gradient Descents,” *IEEE Transactions on Image Processing*, vol. 30, pp. 1044–1056, 2021.

[4] Y. Gao, S. S. Vedula, C. E. Reiley, N. Ahmidi, B. Varadarajan, H. C. Lin, L. Tao, L. Zappella, B. Béjar, D. D. Yuh, C. C. G. Chen, R. Vidal, S. Khudanpur, and G. D. Hager, “JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS): A Surgical Activity Dataset for Human Motion Modeling,” 2014.

[5] L. Xiong, X. Chen, and J. Schneider, “Direct Robust Matrix Factorizatoin for Anomaly Detection,” in *Proc. IEEE International Conference on Data Mining*, 2011, pp. 844–853.

[6] T. M. Press, “Introduction to Machine Learning, Second Edition,” *The MIT Press.*, 5 2019.

[7] O. Simeone, “A Very Brief Introduction to Machine Learning With Applications to Communication Systems,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648–664, Dec. 2018.

[8] A. Marban, V. Srinivasan, W. Samek, J. Fernández, and A. Casals, “Estimation of Interaction Forces in Robotic Surgery using a Semi-Supervised Deep Neural Network Model,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 761–768.

[9] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep Learning for IoT Big Data and Streaming Analytics: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2923–2960, Fourthquarter 2018.

[10] T. Osa, N. Sugita, and M. Mitsuishi, "Online Trajectory Planning and Force Control for Automation of Surgical Tasks," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 675–691, April 2018.

[11] W. Xu, J. Chen, H. Y. Lau, and H. Ren, "Data-driven Methods Towards Learning the Highly Nonlinear Inverse Kinematics of Tendon-driven Surgical Manipulators," *Wiley International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 13, no. 3, p. e1774, Sept. 2017.

[12] H. Mayer, F. Gomez, D. Wierstra, I. Nagy, A. Knoll, and J. Schmidhuber, "A System for Robotic Heart Surgery that Learns to Tie Knots Using Recurrent Neural Networks," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2006, pp. 543–548.

[13] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[15] S. Klanke, S. Vijayakumar, and S. Schaal, "A library for locally weighted projection regression," *Journal of Machine Learning Research*, vol. 9, no. 21, pp. 623–626, 2008. [Online]. Available: http://jmlr.org/papers/v9/klanke08a.html

[16] T. D. Bui, C. V. Nguyen, and R. E. Turner, "Streaming Sparse Gaussian Process Approximations," in *NIPS*, Dec 4-9 2017.

[17] A. Douik, *Riemannian Optimization for Convex and Non-Convex Signal Processing and Machine Learning Applications*. California Institute of Technology, 2020.

[18] Y. Tian, Y. Zhang, and H. Zhang, "Recent Advances in Stochastic Gradient Descent in Deep Learning," *Mathematics*, vol. 11, no. 3, p. 682, 2023.

[19] C. E. Reiley, E. Plaku, and G. D. Hager, "Motion Generation of Robotic Surgical Tasks: Learning from Expert Demonstrations," in *Proc. Annual International Conference of the IEEE Engineering in Medicine and Biology*, Aug. 2010, pp. 967–970.

[20] F. Boabang, R. Glitho, H.Elbiaze, F.Belqami, and O.Alfandi., "A Framework for Predicting Haptic Feedback in 5G Remote Robotic Surgery," in *Proc. IEEE Consumer Communications Networking Conference (CCNC)*, 2020, pp. 1–6.

[21] N. J. Kong, T. K. Stephens, and T. M. Kowalewski, "Da Vinci Tool Torque Mapping over 50,000 Grasps and Its Implications on Grip Force Estimation Accuracy," in *Proc. IEEE International Symposium on Medical Robotics (ISMR)*, March 2018, pp. 1–6.

[22] C. Pacchierotti, M. Abayazid, S. Misra, and D. Prattichizzo, "Teleoperation of steerable flexible needles by combining kinesthetic and vibratory feedback," *IEEE Transactions on Haptics*, vol. 7, no. 4, pp. 551–556, Oct 2014.

[23] J. v. d. D. J. J. de Jong, L. Tonke ; Dankelman, "Dataset on force measurements of needle insertions into two ex-vivo human livers," in *Mendeley Data, v2*, 2017.

[24] M. Babaiasl, F. Yang, and J. P. Swensen, "Towards Water-Jet Steerable Needles," in *Proc. IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*, Aug. 2018, pp. 601–608.

[25] F. Boabang, A. Ebrahimzadeh, R. H. Glitho, H. Elbiaze, M. Maier, and F. Belqasmi, "A machine learning framework for handling delayed/lost packets in tactile internet remote robotic surgery," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4829–4845, 2021.

[26] Y. Ni, J. Sun, X. Yuan, S. Yan, and L. Cheong, "Robust Low-Rank Subspace Segmentation with Semidefinite Guarantees," *Proc. IEEE International Conference on Data Mining Workshops*, pp. 1179–1188, 2010.

[27] E. Kim, S. Choi, and S. Oh, "Structured Low-rank Matrix Approximation in Gaussian Process Regression for Autonomous Robot Navigation," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 69–74.

[28] J. Chen, D. Zhou, Y. Tang, Z. Yang, Y. Cao, and Q. Gu, "Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, ser. IJCAI'20, 2021.

[29] H. Mittal, K. Pandey, and Y. Kant, "ICLR Reproducibility Challenge Report (Padam : Closing The Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks)," *ArXiv*, vol. abs/1901.09517, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:249647677

[30] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *ICLR*, vol. abs/1412.6980, 2015.

[31] X. Chen, S. Liu, R. Sun, and M. Hong, “On the convergence of a class of adam-type algorithms for non-convex optimization,” *arXiv preprint arXiv:1808.02941*, 2018.

[32] D. Zhou, J. Chen, Y. Cao, Y. Tang, Z. Yang, and Q. Gu, “On the convergence of adaptive gradient methods for nonconvex optimization,” *arXiv preprint arXiv:1808.05671*, 2018.

[33] “https://github.com/boabangf21/Proposed_ASGD-Improved-Adam-and-Amsgrad-.”

[34] X. P. Li, L. Huang, H. C. So, and B. Zhao, “A Survey on Matrix Completion: Perspective of Signal Processing,” *arXiv: Signal Processing*, 2019.

[35] S. D. Bopardikar and G. S. E. Ekladious, “Sequential Randomized Matrix Factorization for Gaussian Processes,” in *Proc. IEEE International Conference on Big Data (Big Data)*, 2016, pp. 3957–3959.

[36] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O’Neil, “Fast Direct Methods for Gaussian Processes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 252–265, Feb. 2016.

[37] F. Wu, T. Zhang, L. Li, Y. Huang, and Z. Peng, “RPCANet: Deep Unfolding RPCA Based Infrared Small Target Detection,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 4809–4818.

[38] Y. Li, Y. Yu, Q. Zhang, C. Liang, P. He, W. Chen, and T. Zhao, “Losparse: Structured compression of large language models based on low-rank and sparse approximation,” in *International Conference on Machine Learning*. PMLR, 2023, pp. 20 336–20 350.

[39] J. Zhuang, T. M. Tang, Y. Ding, S. C. Tatikonda, N. C. Dvornek, X. Papademetris, and J. S. Duncan, “Adabelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients,” *ArXiv*, vol. abs/2010.07468, 2020.

[40] S. J. Reddi, S. Kale, and S. Kumar, “On the Convergence of Adam and Beyond,” *ArXiv*, vol. abs/1904.09237, 2018.

[41] H. Zhong, Z. Chen, C. Qin, Z. Huang, V. W. Zheng, T. Xu, and E. Chen, “Adam revisited: a weighted past gradients perspective,” *Frontiers of Computer Science*, vol. 14, pp. 1–16, 2020.

[42] Y. Zhou, K. Huang, C. Cheng, X. Wang, A. Hussain, and X. Liu, "Fastadabelief: Improving Convergence Rate for Belief-Based Adaptive Optimizers by Exploiting Strong Convexity," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2022.

[43] X. Huang, R. Xu, H. Zhou, Z. Wang, Z. Liu, and L. Li, "ACMo: Angle-Calibrated Moment Methods for Stochastic Optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, 2021, pp. 7857–7864.

[44] J. Chen, C. Wolfe, Z. Li, and A. Kyrillidis, "Demon: Improved Neural Network Training With Momentum Decay," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 3958–3962.

[45] K. Verma and A. Maiti, "Wsagrad: a novel adaptive gradient based method," *Applied Intelligence*, vol. 53, no. 11, pp. 14 383–14 399, 2023.

[46] F. Huang, J. Li, and H. Huang, "Super-adam: faster and universal framework of adaptive gradients," *Advances in Neural Information Processing Systems*, vol. 34, pp. 9074–9085, 2021.

[47] A. Chowriappa, R. Wirz, A. R. Ashammagari, and Y. W. Seo, "Prediction from Expert Demonstrations for Safe Tele-surgery," *International Journal of Automation and Computing*, vol. 10, no. 6, pp. 487–497, Dec. 2013.

[48] C. Shin, P. W. Ferguson, S. A. Pedram, J. Ma, E. P. Dutson, and J. Rosen, "Autonomous Tissue Manipulation via Surgical Robot Using Learning Based Model Predictive Control," in *Proc. International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 3875–3881.

[49] N. N., S. J.A., A. M.G., F. F., and M. L.A., "Optimizing Robotic Automatic Suturing Through VR-Enhanced Data Generation for Reinforcement Learning Algorithms," in *2024 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR)*, 2024, pp. 375–383.

[50] E. Sneath, C. Korte, and G. Schaffner, *Semi-Autonomous Robotic Surgery for Space Exploration Missions*.

[51] B. Thananjeyan, A. Garg, S. Krishnan, C. Chen, L. Miller, and K. Goldberg, "Multilateral Surgical pattern Cutting in 2D Orthotropic Gauze with Deep Reinforcement Learning Policies for Tensioning," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2371–2378.

[52] L. Fichera, D. Pardo, and L. S. Mattos, "Modeling Tissue Temperature Dynamics during laser Exposure," in *Proc. International Work-Conference on Artificial Neural Networks (IWANN)*, 2013, pp. 96–106.

[53] H. Huynhnguyen and U. A. Buy, "Toward Gesture Recognition in Robot-Assisted Surgical Procedures," in *2020 2nd International Conference on Societal Automation (SA)*, 2021, pp. 1–4.

[54] B. Béjar Haro, L. Zappella, and R. Vidal, "Surgical gesture classification from video data," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2012: 15th International Conference, Nice, France, October 1-5, 2012, Proceedings, Part I 15*. Springer, 2012, pp. 34–41.

[55] F. Luongo, R. Hakim, J. H. Nguyen, A. Anandkumar, and A. J. Hung, "Deep learning-based computer vision to recognize and classify suturing gestures in robot-assisted surgery," *Surgery*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:221340765

[56] P. Selvam *et al.*, "A deep learning framework for surgery action detection," in *Deep Learning in Personalized Healthcare and Decision Support*. Elsevier, 2023, pp. 315–328.

[57] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, Fourthquarter 2015.

[58] Y. Xu, F. Yin, W. Xu, J. Lin, and S. Cui, "Wireless Traffic Prediction with Scalable Gaussian Process: Framework, Algorithms, and Verification," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1291–1306, 2019.

[59] X. Ge, R. Zhou, and Q. Li, "5G NFV-Based Tactile Internet for Mission-Critical IoT Services," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[60] G. P. Fettweis, "The Tactile Internet: Applications and Challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, March 2014.

[61] N. Promwongsa, A. Ebrahimzadeh, D. Naboulsi, S. Kianpisheh, F. Belqasmi, R. Glitho, N. Crespi, and O. Alfandi, "A Comprehensive Survey of the Tactile Internet: State-of-the-art and Research Directions," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 472–523, Firstquarter 2021.

[62] M. Maier, M. Chowdhury, B. P. Rimal, and D. P. Van, "The Tactile Internet: Vision, Recent progress, and Open Challenges," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 138–145, May 2016.

[63] A. Ebrahimzadeh and M. Maier, "Delay-Constrained Teleoperation Task Scheduling and Assignment for Human+Machine Hybrid Activities Over Fiwi Enhanced Networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1840–1854, Dec. 2019.

[64] H. Beyranvand, M. Lévesque, M. Maier, J. A. Salehi, C. Verikoukis, and D. Tipper, "Toward 5G: FiWi Enhanced LTE-A HetNets With Reliable Low-Latency Fiber Backhaul Sharing and WiFi Offloading," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 690–707, Apr. 2017.

[65] M. Simsek, A. Aijaz, M. Dohler, J. Sachs, and G. Fettweis, "5G-Enabled Tactile Internet," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 460–473, March 2016.

[66] E. Steinbach, S. Hirche, M. Ernst, F. Brandi, R. Chaudhari, J. Kammerl, and I. Vittorias, "Haptic Communications," *Proceedings of the IEEE*, vol. 100, no. 4, pp. 937–956, 2012.

[67] K. Antonakoglou, X. Xu, E. Steinbach, T. Mahmoodi, and M. Dohler, "Toward Haptic Communications over the 5G Tactile Internet," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3034–3059, Fourth quarter 2018.

[68] C. She, C. Sun, Z. Gu, Y. Li, C. Yang, H. V. Poor, and B. Vucetic, "A Tutorial on Ultrareliable and Low-Latency Communications in 6G: Integrating Domain Knowledge Into Deep Learning," *Proceedings of the IEEE*, vol. 109, no. 3, pp. 204–246, 2021.

[69] D. Nguyen-Tuong, J. Peters, and M. Seeger, "Local Gaussian Process Regression for Real Time Online Model Learning and Control," in *Proc. ACM International Conference on Neural Information Processing Systems*, ser. NIPS'08. Red Hook, NY, USA: Curran Associates Inc., 2008, p. 1193–1200.

[70] C. E. Rasmussen, *Gaussian Processes in Machine Learning*. Springer Berlin Heidelberg, 2004.

[71] D. Nguyen-Tuong and J. Peters, "Model Learning for Robot Control: A Survey," *Cognitive Processing*, vol. 12, pp. 319–340, 2011.

[72] T. Zhou and D. Tao, "GoDec: Randomized Low-rank Sparse Matrix Decomposition in Noisy Case," in *Proc. International Conference on Machine Learning (ICML)*, 2011, pp. 33–40.

[73] X. Yu, T. Liu, X. Wang, and D. Tao, "On Compressing Deep Models by Low Rank and Sparse Decomposition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 67–76.

[74] K. Guo, X. Xu, and D. Tao, “Discriminative GoDec+ for Classification,” *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3414–3429, 2017.

[75] W. Shi, X. Zhang, M. Sun, X. Zou, Y. Wei, and G. Min, “Deep Neural Network based Monaural Speech Enhancement with Sparse and Low-rank Decomposition,” in *Proc. IEEE International Conference on Communication Technology (ICCT)*, 2017, pp. 1644–1647.

[76] O. Solomon, R. Cohen, Y. Zhang, Y. Yang, Q. He, J. Luo, R. J. G. van Sloun, and Y. C. Eldar, “Deep Unfolded Robust PCA With Application to Clutter Suppression in Ultrasound,” *IEEE Transactions on Medical Imaging*, vol. 39, no. 4, pp. 1051–1063, 2020.

[77] Q. Wang, Q. Gao, G. Sun, and C. Ding, “Double robust principal component analysis,” *Neurocomputing*, vol. 391, pp. 119–128, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220301557

[78] B. Bao, G. Liu, C. Xu, and S. Yan, “Inductive Robust Principal Component Analysis,” *IEEE Transactions on Image Processing*, vol. 21, no. 8, pp. 3794–3800, 2012.

[79] F. Despinoy, D. Bouget, G. Forestier, C. Penet, N. Zemiti, P. Poignet, and P. Jannin, “Unsupervised Trajectory Segmentation for Surgical Gesture Recognition in Robotic Training,” *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 6, pp. 1280–1291, June 2016.

[80] M. Brand, “Incremental Singular Value Decomposition of Uncertain Data with Missing Values,” in *Proc. European Conference on Computer Vision (ECCV)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 707–720.

[81] R. Gupta, S. Tanwar, S. Tyagi, and N. Kumar, “Tactile-Internet-based Telesurgery System for Healthcare 4.0: An Architecture, Research Challenges, and Future Directions,” *IEEE Network*, vol. 33, no. 6, pp. 22–29, Nov. 2019.

[82] H. Liu, Y. Ong, X. Shen, and J. Cai, “When Gaussian Process Meets Big Data: A Review of Scalable GPs,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4405–4423.

[83] A. Ruszczynski, *Nonlinear Optimization*. USA: Princeton University Press, 2006.

[84] X. Li, K. Xie, X. Wang, G. Xie, D. Xie, Z. Li, J. Wen, Z. Diao, and T. Wang, “Quick and Accurate False Data Detection in Mobile Crowd Sensing,” *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1339–1352, June 2020.

[85] S. Mattisson, "Overview of 5G Requirements and Future Wireless Networks," in *Proc. IEEE European Solid State Circuits Conference (ESSCIRC)*, 2017, pp. 1–6.

[86] Y. Zhang and Z. Lu, "Penalty Decomposition Methods for Rank Minimization," in *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., 2011, pp. 46–54.

[87] D. Alpay, F. Colombo, D. Kimsey, and I. Sabadini, "The Spectral Theorem for Unitary Operators Based on the S-Spectrum," *Milan Journal of Mathematics*, vol. 84, pp. 41–61, June 2016.

[88] F. Aurzada, M. Lévesque, M. Maier, and M. Reisslein, "Fiwi Access Networks Based on Next-Generation PON and Gigabit-Class WLAN Technologies: A Capacity and Delay Analysis," *IEEE/ACM Transactions on Networking*, vol. 22, no. 4, pp. 1176–1189, 2014.

[89] M. Maier and A. Ebrahimzadeh, "Towards Immersive Tactile Internet Experiences: Low-latency FiWi enhanced Mobile Networks with Edge Intelligence (Invited)," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 11, no. 4, pp. B10–B25, Apr. 2019.

[90] E. Steinbach, S. Hirche, M. Ernst, F. Brandi, R. Chaudhari, J. Kammerl, and I. Vittorias, "Haptic Communications," *Proceedings of the IEEE*, vol. 100, no. 4, pp. 937–956, 2012.

[91] B. van Amsterdam, M. J. Clarkson, and D. Stoyanov, "Gesture Recognition in Robotic Surgery: A Review," *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 6, pp. 2021–2035, 2021.

[92] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 1929–1958, jan 2014.

[93] J. Chen and Q. Gu, "Padam: Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks," 2019. [Online]. Available: https://openreview.net/forum?id=BJll6o09tm

[94] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding Data Augmentation for Classification: When to Warp?" in *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2016, pp. 1–6.

[95] L. Chen, H. Fei, Y. Xiao, J. He, and H. Li, "Why batch normalization works? a buckling perspective," in *2017 IEEE International Conference on Information and Automation (ICIA)*, 2017, pp. 1184–1189.

[96] F. Y. Wu and F. Tong, "Non-Uniform Norm Constraint LMS Algorithm for Sparse System Identification," *IEEE Communications Letters*, vol. 17, no. 2, pp. 385–388, 2013.

[97] J. Duchi, E. Hazan, and Y. Singer, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,” *Journal of Machine Learning Research*, vol. 12, no. null, p. 2121–2159, jul 2011.

[98] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, “Convolutional lstm network: A machine learning approach for precipitation nowcasting,” *Advances in neural information processing systems*, vol. 28, 2015.

[99] F. Karim, S. Majumdar, H. Darabi, and S. Chen, “LSTM fully convolutional networks for time series classification,” *IEEE Access*, vol. 6, pp. 1662–1669, 2018.

[100] T. Szandala, “Review and comparison of commonly used activation functions for deep neural networks,” *Springer Bio-inspired Neurocomputing*, pp. 203–224, 2020.

[101] H.-X. Wen, S.-Q. Yang, Y.-Q. Hong, and H. Luo, “A Partial Update Adaptive Algorithm for Sparse System Identification,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 240–255, 2020.

[102] F. Wu and F. Tong, “Gradient optimization p-norm-like constraint lms algorithm for sparse system estimation,” *Elsevier Signal Processing*, vol. 93, no. 4, pp. 967–971, 2013.

[103] P. Xue and B. Liu, “Adaptive equalizer using finite-bit power-of-two quantizer,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 6, pp. 1603–1611, 1986.

[104] M. He, Y. Liang, J. Liu, and D. Xu, “Convergence of adam for non-convex objectives: Relaxed hyperparameters and non-ergodic case,” *ArXiv*, vol. abs/2307.11782, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:260125579

[105] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:18268744