Abhishek

thesis_v5_final_rmoved_AI.pdf

Delhi Technological University

Document Details

Submission ID

trn:oid:::27535:98622730

Submission Date

May 31, 2025, 3:03 AM GMT+5:30

Download Date

Nov 19, 2025, 2:56 PM GMT+5:30

File Name

thesis_v5_final_rmoved_AI.pdf

File Size

508.6 KB

45 Pages

11,431 Words

68,513 Characters

18% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text

Match Groups

188 Not Cited or Quoted 18%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

10%	Internet sources
8%	Publications
16%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

188 Not Cited or Quoted 18%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

10%	Internet sources
8%	Publications
16%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	dspace.dtu.ac.in:8080	2%
2	Internet	arxiv.org	<1%
3	Internet	www.dspace.dtu.ac.in:8080	<1%
4	Submitted works	Chester College of Higher Education on 2024-10-10	<1%
5	Submitted works	University of Sydney on 2025-05-25	<1%
6	Publication	"Pattern Recognition", Springer Science and Business Media LLC, 2025	<1%
7	Submitted works	University of Essex on 2025-04-17	<1%
8	Submitted works	Abu Dhabi University on 2024-12-01	<1%
9	Submitted works	University of Wollongong on 2025-05-28	<1%
10	Publication	R. N. V. Jagan Mohan, B. H. V. S. Rama Krishnam Raju, V. Chandra Sekhar, T. V. K. P...	<1%

11	Internet	www.frontiersin.org	<1%
12	Internet	www.mdpi.com	<1%
13	Submitted works	Leiden University on 2024-11-12	<1%
14	Publication	Leong, Su Yi. "Little Drops Make the Mighty Ocean: The Influence of Collectivism i...	<1%
15	Submitted works	University of Lancaster on 2025-04-07	<1%
16	Submitted works	Queen Mary and Westfield College on 2025-05-06	<1%
17	Submitted works	University of Northampton on 2025-05-18	<1%
18	Internet	export.arxiv.org	<1%
19	Submitted works	University of Exeter on 2025-03-29	<1%
20	Submitted works	Delhi Technological University on 2025-05-02	<1%
21	Publication	Korrapati, Bhuvana. "Enhancing Federated Learning Efficiency Through Dynamic ...	<1%
22	Submitted works	Cornell University on 2025-05-02	<1%
23	Submitted works	Sheffield Hallam University on 2024-09-04	<1%
24	Submitted works	University of Sydney on 2024-05-24	<1%

25	Internet	napier-repository.worktribe.com	<1%
26	Submitted works	Abra State Institute of Sciences and Technology on 2024-07-10	<1%
27	Publication	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Intelli...	<1%
28	Publication	Gao, Yunhe. "Towards Generalist Medical Artificial Intelligence: Model, Data, Kno...	<1%
29	Submitted works	University College London on 2024-04-19	<1%
30	Internet	peerj.com	<1%
31	Submitted works	Roehampton University on 2024-08-27	<1%
32	Publication	Gao, Zhidong. "Ensuring Resource-Efficiency and Trustworthiness in Machine Lea...	<1%
33	Submitted works	University of Glasgow on 2025-04-12	<1%
34	Submitted works	University of Northumbria at Newcastle on 2025-05-22	<1%
35	Internet	ses.library.usyd.edu.au	<1%
36	Internet	www.geeksforgeeks.org	<1%
37	Internet	www.nature.com	<1%
38	Submitted works	UNIBA on 2024-10-14	<1%

39	Internet	link.springer.com	<1%
40	Submitted works	Loughborough University on 2025-05-01	<1%
41	Submitted works	Obudai Egyetem on 2025-05-19	<1%
42	Submitted works	University of Newcastle upon Tyne on 2025-05-30	<1%
43	Submitted works	University of Wollongong on 2025-03-26	<1%
44	Submitted works	Australian National University on 2025-05-23	<1%
45	Submitted works	Nanyang Technological University on 2025-05-18	<1%
46	Internet	publikationen.reutlingen-university.de	<1%
47	Publication	Alkhaldi, Eid. "Ensemble Optimization for Histological Image Classication", The U...	<1%
48	Submitted works	Coventry University on 2025-04-10	<1%
49	Submitted works	Higher Education Commission Pakistan on 2024-10-28	<1%
50	Submitted works	University of Lincoln on 2021-03-07	<1%
51	Submitted works	University of Sheffield on 2025-01-28	<1%
52	Internet	docshare.tips	<1%

53	Internet	medium.com	<1%
54	Internet	pdfcookie.com	<1%
55	Submitted works	Australian National University on 2025-05-20	<1%
56	Submitted works	Keimyung University on 2024-05-17	<1%
57	Submitted works	Universiti Putra Malaysia on 2016-12-01	<1%
58	Submitted works	University of Melbourne on 2024-10-25	<1%
59	Submitted works	University of Ulster on 2025-04-22	<1%
60	Internet	codeberg.org	<1%
61	Internet	dspace.daffodilvarsity.edu.bd:8080	<1%
62	Internet	huggingface.co	<1%
63	Internet	i-rep.emu.edu.tr:8080	<1%
64	Publication	"Natural Language Processing and Chinese Computing", Springer Science and Bu...	<1%
65	Submitted works	Coventry University on 2025-04-10	<1%
66	Publication	Cummings, Aaron. "Closed Domain Question Answering with Language Models: A...	<1%

67	Publication	Kim, Youngeun. "Algorithmic Approaches for Empowering Spike-Based Machine I...	<1%
68	Submitted works	King's College on 2025-04-01	<1%
69	Submitted works	Queen's University of Belfast on 2025-04-25	<1%
70	Publication	Sang-Hyun Cho, Dohyun Kim, Hyuk-Chul Kwon, Minho Kim. "Exploring the potenti...	<1%
71	Submitted works	University of Birmingham on 2025-04-23	<1%
72	Submitted works	University of Surrey on 2024-05-17	<1%
73	Submitted works	University of Wollongong on 2023-12-15	<1%
74	Submitted works	City University of Hong Kong on 2023-10-13	<1%
75	Submitted works	City University of Hong Kong on 2024-11-26	<1%
76	Submitted works	Columbia University on 2024-05-10	<1%
77	Publication	Fakorede, Olukorede Joseph. "Techniques for Enhancing Adversarial Robustness ...	<1%
78	Publication	J. A. Benediktsson. "Conjugate-gradient neural networks in classification of multi...	<1%
79	Submitted works	Loughborough University on 2024-09-25	<1%
80	Submitted works	Sheffield Hallam University on 2025-01-16	<1%

81	Submitted works	South Bank University on 2025-03-21	<1%
82	Submitted works	University of Newcastle on 2024-12-06	<1%
83	Submitted works	University of Southern Queensland on 2025-05-01	<1%
84	Submitted works	University of Warwick on 2025-03-01	<1%
85	Internet	ace.ewapublishing.org	<1%
86	Internet	aclanthology.org	<1%
87	Submitted works	islingtoncollege on 2025-05-11	<1%
88	Internet	soka.repo.nii.ac.jp	<1%
89	Internet	www.ijisae.org	<1%
90	Internet	www.microsoft.com	<1%
91	Submitted works	Aston University on 2024-05-07	<1%
92	Publication	Jiaxing Liu, Chaofeng Sha, Xin Peng. "An Empirical Study of Parameter-Efficient Fi...	<1%
93	Publication	Khan, Muhammad Osama. "Towards Efficient and Responsible AI: Developing Res...	<1%
94	Submitted works	Liverpool John Moores University on 2021-01-29	<1%

95 Submitted works

Liverpool John Moores University on 2023-03-14 <1%

96 Submitted works

Liverpool John Moores University on 2024-08-11 <1%

97 Publication

R. Prajwal, S. J. Pawan, Shahin Nazarian, Nicholas Heller, Christopher J. Weight, Vi... <1%

98 Publication

Sanger, Mario. "Representation Learning for Biomedical Text Mining", Humboldt ... <1%

99 Submitted works

The Robert Gordon University on 2025-04-24 <1%

100 Submitted works

University of Stirling on 2023-04-21 <1%

101 Submitted works

University of Sydney on 2025-05-21 <1%

102 Submitted works

University of Warwick on 2024-10-02 <1%

103 Submitted works

University of York on 2024-04-24 <1%

104 Internet

ar5iv.labs.arxiv.org <1%

105 Internet

deepai.org <1%

106 Internet

dokumen.pub <1%

107 Internet

download.bibis.ir <1%

108 Internet

elibrary.tucl.edu.np <1%

109	Internet	escholarship.org	<1%
110	Internet	open.metu.edu.tr	<1%
111	Internet	research-information.bris.ac.uk	<1%
112	Internet	scholar.uoc.ac.in	<1%
113	Submitted works	King's College on 2024-08-06	<1%
114	Submitted works	The University of the West of Scotland on 2024-12-09	<1%
115	Submitted works	University of Witwatersrand on 2024-12-26	<1%
116	Submitted works	CSU, San Jose State University on 2025-05-10	<1%
117	Publication	H.L. Gururaj, Francesco Flammini, S. Srividhya, M.L. Chayadevi, Sheba Selvam. "Co...	<1%
118	Submitted works	Imperial College of Science, Technology and Medicine on 2019-06-11	<1%
119	Submitted works	Khalifa University of Science Technology and Research on 2024-07-03	<1%
120	Submitted works	King Faisal University on 2025-05-16	<1%
121	Submitted works	Liverpool John Moores University on 2023-12-14	<1%
122	Submitted works	National Taiwan University on 2022-08-19	<1%

123 Publication

Thangaprakash Sengodan, Sanjay Misra, M Murugappan. "Advances in Electrical ... <1%

124 Submitted works

University College London on 2024-09-11 <1%

125 Submitted works

University of Bristol on 2025-03-20 <1%

126 Submitted works

University of Stirling on 2024-05-25 <1%

Efficient Operator Chain Recognition via Low-Rank Adaptation in Deep Learning Frameworks

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
ARTIFICIAL INTELLIGENCE

Submitted by

ABHISHEK MAURYA (23/AFI/20)

Under the supervision of

Ms. Anukriti Kaushal

COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

JUNE, 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, **Abhishek Maurya**, Roll No. **23/AFI/20**, student of M.Tech (**Artificial Intelligence**), hereby declare that the Project Dissertation titled "**Efficient Operator Chain Recognition via Low-Rank Adaptation in Deep Learning Frameworks**" submitted by me to the **Department of Computer Science and Engineering**, Delhi Technological University, Delhi, in partial fulfilment of the requirements for the award of the degree of Master of Technology, is my original work. This dissertation has not been copied from any source without proper citation, nor has it previously been submitted for the award of any Degree, Diploma, Associateship, Fellowship, or any other similar title or recognition.

Place: Delhi

Abhishek Maurya

Date:

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled **"Efficient Operator Chain Recognition via Low-Rank Adaptation in Deep Learning Frameworks"**, submitted by **Abhishek Maurya, Roll No. 23/AFI/20**, from the Computer Science and Engineering department at Delhi Technological University, Delhi, is a record of the project work carried out by the student under my supervision. To the best of my knowledge, this work has not been submitted, either in part or in full, for any degree or diploma at this University or elsewhere.

Place: Delhi	**Ms. Anukriti Kaushal**
Date:	**SUPERVISOR**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I would like to sincerely thank **Ms. Anukriti Kaushal** (Assistant Professor) for her continuous guidance and support throughout this project. She clearly outlined the tasks I needed to complete and took the time to explain the importance and real-world relevance of the project, which truly motivated me to achieve my goals. Whenever I faced challenges, she was always ready to help and patiently answered all my questions. This project's success wouldn't have been possible without her constant encouragement and invaluable support.

125

112

Place: Delhi Abhishek Maurya

Date:

Abstract

This thesis presents an efficient approach for recognizing operator chains in deep learning frameworks using Low-Rank Adaptation (LoRA). The increasing complexity of deep learning models has created significant computational demands, making efficient adaptation crucial for deployment in various environments. We propose a novel application of LoRA to fine-tune Vision Transformers (ViTs) for recognizing operator chains with minimal parameter updates.

Our approach achieves 92.23% accuracy while fine-tuning only 0.35% of the model parameters, demonstrating both computational efficiency and high recognition performance. We implement various optimization techniques, including mixed precision training, gradient accumulation, and early stopping to enhance both training efficiency and model performance. Our experimental results confirm that LoRA enables a significant reduction in trainable parameters while maintaining competitive performance, making it suitable for resource-constrained environments and preserving pre-trained knowledge. The framework's adaptability makes it applicable across various sequence recognition tasks beyond operator chains.

20

Contents

54

Candidate's Declaration i

Certificate ii

Acknowledgement iii

Abstract iv

Content vi

List of Tables vii

List of Figures viii

1 INTRODUCTION ix
- 1.1 Overview . . . ix
- 1.2 Motivation . . . ix
- 1.3 Problem Statement . . . x
- 1.4 Objectives and Contributions . . . x
- 1.5 Thesis Structure . . . 1

2 LITERATURE REVIEW 2
- 2.1 Overview . . . 2
- 2.2 Related Work . . . 2
  - 2.2.1 Operator Chain Recognition Techniques . . . 2
  - 2.2.2 Methods of Convolutional Neural Networks . . . 3
  - 2.2.3 Vision Transformers . . . 4
  - 2.2.4 Parameter-Efficient Fine-Tuning . . . 4
  - 2.2.5 Low-Rank Adaptation . . . 4
- 2.3 Integration of Features . . . 5
- 2.4 Comparative Study of Image Manipulation and Operator Chain Detection 5
  - 2.4.1 Challenges in Continuous Recognition . . . 5
  - 2.4.2 General-Purpose Image Manipulation Detectors . . . 6
  - 2.4.3 Operator Chain Detection Techniques . . . 6
  - 2.4.4 Performance and Robustness Summary . . . 6
  - 2.4.5 Discussion . . . 6

3 METHODOLOGY 8
- 3.1 Data Preparation . . . 8
  - 3.1.1 Dataset Description . . . 8
  - 3.1.2 Data Collection and Preprocessing . . . 10

52

66

14

65

48

50

1

79

3.2 Model Architecture . . . 11
3.2.1 Vision Transformer Base Model . . . 11
3.2.2 Low-Rank Adaptation[1] Implementation . . . 12
3.3 Training Strategy . . . 13
3.3.1 Mixed Precision Training . . . 13
3.3.2 Gradient Accumulation . . . 15
3.3.3 Early Stopping . . . 15
3.3.4 Optimization Parameters . . . 16
3.3.5 Performance Metrics . . . 16
3.4 Implementation Details . . . 17
3.4.1 Hardware and Software Setup . . . 17
3.4.2 Model Initialization . . . 17
3.4.3 Model Architecture Flow Diagram . . . 18
3.4.4 ViT Transformer Block with LoRA Injection . . . 19
3.4.5 Data Pipeline . . . 19
3.5 Evaluation Methodology . . . 20
3.5.1 Metrics . . . 20
3.5.2 Ablation Studies . . . 20
3.5.3 Comparison with Baselines . . . 21

4 RESULTS AND ANALYSIS 22
4.1 Introduction to Results and Analysis . . . 22
4.1.1 Brief Overview of the Results and Their Significance . . . 22
4.2 Training Dynamics . . . 22
4.2.1 Training Dynamics and Observations . . . 22
4.3 Performance Evaluation . . . 24
4.3.1 Per-Class Performance Analysis . . . 24
4.4 Parameter Efficiency . . . 25
4.5 Comparison with Baseline Methods . . . 26
4.6 Ablation Studies . . . 27
4.6.1 Impact of LoRA Rank . . . 27
4.6.2 Impact of Target Modules . . . 28
4.6.3 Impact of Training Strategies . . . 28
4.7 Discussion . . . 29

5 CONCLUSION AND FUTURE SCOPE 30
5.1 Summary of Contributions . . . 30
5.2 Limitations . . . 30
5.3 Future Research Directions . . . 31
5.4 Broader Implications . . . 31
5.5 Concluding Remarks . . . 32

79

List of Tables

2.1 Comparison of General-Purpose Manipulation Detectors 6
2.2 Comparison of Operator Chain Detection Methods 6
2.3 Capability Matrix for Reviewed Models 6

3.1 Vision Transformer Architecture Details 13
3.2 Breakdown of Trainable Parameters with LoRA 13

4.1 Epoch-wise Training Performance Metrics 23
4.2 Final Test Performance Metrics 24
4.3 Per-Class Performance Metrics 26
4.4 Comparison of Different Parameter-Efficient Fine-Tuning Methods 27
4.5 Impact of Target Modules on Model Performance 28

List of Figures

3.1 Overview of the training and evaluation process of the Vision Transformer (ViT) enhanced with Low-Rank Adaptation (LoRA). The pipeline shows key stages such as input data processing, LoRA-based fine-tuning, and model validation to measure performance. 18
3.2 Transformer Block with LoRA injection in the Query and Value projections. 19

4.1 Training and validation loss curves over epochs, showing the model's convergence pattern and the point where early stopping was triggered after epoch 7. 23
4.2 Confusion matrix for test set predictions, showing the model's performance across all 10 classes. 25
4.3 Comparison of trainable parameters: LoRA vs. full fine-tuning, highlighting the significant parameter reduction achieved with LoRA. 26
4.4 Impact of LoRA rank on model accuracy, showing diminishing returns beyond rank 8. 27

90

Chapter 1

INTRODUCTION

1.1 Overview

Convolution Neural Networks have achieved remarkable success across various domains, yet their growing complexity introduces significant computational and memory constraints, particularly during model fine-tuning for specialized tasks. This challenge is especially evident in operator chain recognition where identifying sequences of operations efficiently can dramatically improve system performance.

119 Traditional fine-tuning approaches updates all of the model parameters, which proves computationally expensive and often results in catastrophic forgetting of knowledge encoded in pre-trained weights. This thesis addresses this challenge by introducing a parameter-efficient approach based on Low-Rank Adaptation(LoRA)[1] to fine-tune pre-trained Vision Transformer (ViT) model for operator chain recognition tasks.

Operator chains represent sequences of operations that frequently occur together in computational graphs, program execution flows, or neural network architectures. Efficiently recognizing these patterns enables various optimizations, including operation fusion, kernel specialization, and execution planning. However, traditional methods for identifying these patterns often rely on rule-based systems or computationally intensive graph analyses that struggle to generalize across different frameworks and hardware platforms.

By reformulating operator chain recognition as a visual pattern recognition problem, we leverage the powerful representation capabilities of Vision Transformers while addressing their computational limitations through Low-Rank Adaptation[1]. This ap-
40 92 proach injects trainable low rank decomposition matrices into the models while keeping
75 most pre-trained weights frozen, significantly reducing the numbers of parameters that need updating during fine-tuning.

103

1.2 Motivation

The primary motivation for conducting this study:

116

1. **Computational Efficiency:** As deep learning models grow in complexity, the computational resources required for fine-tuning become prohibitive. Parameter-efficient adaptation techniques like LoRA[1] address this challenge by significantly reducing the number of trainable parameters.

2. **Preserving Pre-trained Knowledge:** Standard fine-tuning approaches risk catastrophic forgetting of valuable information encoded in pre-trained weights. LoRA[1]

mitigates this risk by maintaining most of the pre-trained parameters frozen.

3. **Adaptability to New Domains:** The increasing diversity of operator patterns across different frameworks and hardware architectures necessitates adaptable recognition systems that can efficiently transfer to new domains with minimal retraining.

4. **Resource-Constrained Deployment:** Many real world applications run in the environments with the limited computational resources, where full-parameter fine-tuning is impractical. Parameter-efficient approaches enable adaptation in these constrained environments.

1.3 Problem Statement

This thesis addresses the following key research question: How can we adapt efficiently the pre-trained Vision Transformer models to recognize operator chains with minimal parameter updates while maintaining high recognition accuracy?

Specifically, we focus on:

1. Developing a parameter-efficient fine-tuning approach using LoRA for Vision Transformers applied to operator chain recognition

2. Evaluating the trade-offs between parameter efficiency, computational requirements, and recognition accuracy

3. Designing optimization strategies to enhance training efficiency and model performance

4. Analyzing the effectiveness of the approach across different types of operator patterns

1.4 Objectives and Contributions

The primary objectives of this thesis include:

1. Implement and evaluate a LoRA-based approach for fine-tuning Vision Transformers on operator chain recognition tasks

2. Develop optimization strategies that enhance training efficiency while maintaining recognition accuracy

3. Provide a comprehensive analysis of the parameter-performance trade-offs

4. Demonstrate the applicability of the approach across different operator pattern categories

Our key contributions include:

1. A novel application of LoRA to operator chain recognition that achieves 92.23% accuracy while fine tuning only 0.35% of model parameters

2. A comprehensive implementation framework integrating mixed precision training, gradient accumulation, and early stopping for efficient fine-tuning

3. Empirical evaluation demonstrating the effectiveness of the approach across different operator pattern types
4. Analysis of the impact of the different LoRA configurations on recognition performance

1.5 Thesis Structure

This thesis's remaining sections are structured as follows:

- Chapter 2 provides a literature survey covering deep learning approaches for operator recognition, Vision Transformers, parameter-efficient fine-tuning techniques, and Low-Rank Adaptation[1].
- Chapter 3 details our methodology, including data preparation, model architecture, LoRA implementation, training strategies, and optimization techniques.
- Chapter 4 displays the findings of the experiment and analysis, comparing our approach with baseline methods and evaluating performance across different operator pattern categories.
- Chapter 5 wraps up the thesis by outlining the main conclusions, talking about the limits, and offering ideas for future study directions.

Chapter 2

LITERATURE REVIEW

2.1 Overview

This chapter presents a comprehensive review of literature relevant to operator chain recognition and parameter-efficient fine-tuning[1][2][3] of deep learning models. We begin by examining traditional approaches to operator pattern recognition, followed by a discussion of Convolution Neural Network methods applied to this domain. We then explore Vision Transformers (ViTs) and their applications, concluding with a detailed review of parameter-efficient fine-tuning techniques, with particular emphasis on Low-Rank Adaptation(LoRA)[1].

2.2 Related Work

2.2.1 Operator Chain Recognition Techniques

Early methods for recognizing operator chains mostly depended on rule-based pattern matching and graph-based algorithms. These techniques relied on static analysis to detect common structures within computational graphs and program execution flows. For instance, Chen et al. [4] introduced TVM, which used graph-based pattern matching to identify and optimize operator chains in deep learning models.

In recent years, machine learning techniques have become increasingly popular for operator pattern recognition. Jia et al. [5] proposed TASO, a graph substitution optimizer that applies reinforcement learning to uncover optimization opportunities within computational graphs. Along similar lines, Zheng et al. [?] developed Ansor, which uses a learning-based search strategy to efficiently optimize tensor program implementations.

Deep learning approaches to operator pattern recognition have gained traction in recent years. Wang et al. [6] introduced PET, a system that uses graph neural networks to identify operator patterns for hardware acceleration. Renda et al. [7] presented DiffTune, which employs differentiable programming to optimize operator implementations through learned parameters.

Despite these advances, most existing approaches still face challenges in terms of computational efficiency, adaptability to new patterns, and ease of deployment in resource-constrained environments.

2.2.2 Methods of Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have become a backbone in digital image forensics due to their ability to automatically learn hierarchical patterns. Their application to manipulation detection has yielded a range of models with varying focuses.

MISLnet

MISLnet, proposed by Bayar and Stamm [8], introduced the concept of a constrained convolutional layer that suppresses content information while amplifying manipulation-related residuals. This was achieved by fixing the first layer to behave like a high-pass filter and training deeper layers to extract manipulation patterns. MISLnet has shown strong performance on uncompressed, high-resolution images and demonstrates the feasibility of learning forensic features directly from data.

RF-CNN

RF-CNN, a lightweight architecture by Singhal et al. [9], utilizes frequency domain features of image residuals, particularly those derived from median filtering. By transforming residuals into the frequency domain before inputting them into a CNN, RF-CNN reduces computational cost while maintaining high manipulation detection accuracy. It performs particularly well on small-resolution and JPEG-compressed images, making it suitable for real-world applications.

FFR-CNN

FFR-CNN by Agarwal and Gupta [10] expands on the idea of residual preprocessing by using four different filters—average, Gaussian, Laplacian, and median. These filtered outputs are stacked to form a volumetric input for a 3D CNN, allowing the model to jointly learn from diverse manipulation signatures. FFR-CNN demonstrates robustness under anti-forensic attacks and JPEG compression, making it effective for passive forensic tasks.

Two-Stream CNN

The Two-Stream CNN approach by Liao et al. [11] introduces a bifurcated network that separately processes tampering artifacts and local noise residuals. This architecture is tailored for operator chain detection, particularly sequences involving two manipulations. Despite its simplicity, the dual-path design effectively isolates different types of forensic evidence, improving chain identification.

ReMReNet

ReMReNet, proposed by Kadha et al. [12], takes a residual learning approach optimized for JPEG compression. It comprises two parallel streams—one focused on compression artifacts using DCT residuals and the other on spatial noise patterns. ReMReNet is designed to detect manipulation chains embedded within JPEG compressed images, a common challenge in real-world scenarios like social media forensics.

2.2.3 Vision Transformers

Dosovitskiy et al. introduced Vision Transformers(ViTs)[13] transformer architecture, which was first created for natural language processing, by modifying it to computer vision tasks. ViTs divide images into patches, linearly embed them, and process them using transformer encoder blocks. Despite requiring large datasets for pre-training, ViTs have achieved state-of-the-art performance on various vision tasks[13].

Several works have extended the original ViT architecture to enhance its performance and efficiency. Touvron et al.[14] introduced DeiT, which employs a teacher-student strategy to distill knowledge into Vision Transformers, reducing the need for large-scale pre-training. Liu et al.[15] presented Swin Transformer, which introduces hierarchical feature maps and shifted windows to improve efficiency and performance on dense prediction tasks.

The application of Vision Transformers to pattern recognition in non-traditional domains remains an area of active research. While ViTs have demonstrated impressive performance on standard image classification tasks, their application to specialized domains like operator chain recognition presents unique challenges and opportunities.

2.2.4 Parameter-Efficient Fine-Tuning

[2][1] Large pre-trained models' fine-tuning has become a standard approach in transfer learning, but it requires updating all model parameters, which is computationally intensive. Several parameter-efficient methods have been suggested to deal with this problem:

1. **Adapter Methods:** Houlsby et al. [2] proposed adding small adapter modules between transformer layers while freezing the original parameters. Pfeiffer et al. [16] extended this approach with AdapterFusion, which combines multiple task-specific adapters.

2. **Prompt Tuning:** Li and Liang[17] presented prefix tuning, a technique that involves appending learnable vectors to the values and keys in transformer layers. Lester et al. [3] simplified this approach to only prepend vectors to the input embeddings.

3. **Low-Rank Adaptation (LoRA):** Hu et al. [1] proposed LoRA, which represents weight updates using low-rank decomposition matrices, significantly reducing the number of trainable parameters.

2.2.5 Low-Rank Adaptation

Low-Rank Adaptation(LoRA)[1] has emerged as a particularly promising approach for parameter-efficient fine tuning. The foundation of LoRA is the finding that during fine-tuning, updates to pre-trained weight matrices frequently have a low intrinsic rank. By exploiting this property, LoRA parameterizes weight updates as:

$$W' = W + \Delta W = W + BA[1] \tag{2.1}$$

where $W \in \mathbb{R}^{d\times k}$ is the pre trained weight matrix, $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ are low-rank matrices (with rank $r \ll \min(d,k)$), and $\Delta W = BA$ is the update to the weight matrix[1].

10 Only the low-rank matrices A and B are trained during fine-tuning, while the pre-trained weights W are frozen. This method preserves model performance while drastically lowering the number of trainable parameters. For instance, LoRA can decrease the number of trainable parameters by orders of magnitude in a standard Vision Transformer with millions of parameters. 49

Several research has shown that LoRA works well in a variety of fields. Hu et al.[1] showed that LoRA achieves competitive performance on natural language processing task while training only 0.5% of the parameters. Wang et al.[18] extended LoRA to mixture-of-experts models with AdaMix, which dynamically combines multiple LoRA modules.

Despite its success, the application of LoRA to vision tasks, particularly in specialized domains like operator chain recognition, remains relatively unexplored. By examining how well LoRA works for optimizing Vision Transformers in the context of operator chain recognition, this thesis seeks to close this gap.

2.3 Integration of Features

Effective operator chain recognition requires integrating multiple types of features, including:

1. **Spatial Features:** Information about the position and arrangement of operators in computational graphs or program flows.

2. **Semantic Features:** Understanding of the meaning and function of different operators and their relationships.

3. **Contextual Features:** Information about the broader context in which operator chains appear, including input/output dependencies and hardware constraints.

114 Vision Transformers excel at integrating these features through their self-attention mechanism, which can capture complex relationships between operators and their context. By treating operator chains as visual patterns, we can leverage the powerful representation capabilities of ViTs while addressing their computational limitations through Low-Rank Adaptation.

2.4 Comparative Study of Image Manipulation and Operator Chain Detection

Recent studies have focused on detecting not only the presence of manipulation in images but also the sequence in which such operations were applied. This section compares deep learning-based techniques against the ViT+LoRA model presented in this thesis, which achieves 92.2% accuracy.

2.4.1 Challenges in Continuous Recognition

Continuous operator chain recognition—identifying operator patterns in streaming data or continuous execution flows—presents additional challenges compared to static recognition:

1. **Temporal Dependencies:** Operator chains may span across time, requiring models to capture temporal dependencies.

2. **Variable Length:** Operator chains can have variable lengths, making it challenging to design fixed-size representations.
3. **Context Sensitivity:** The interpretation of operators may depend on their context, requiring models to maintain contextual information over time.

2.4.2 General-Purpose Image Manipulation Detectors

Table 2.1: Comparison of General-Purpose Manipulation Detectors

Model	**Approach**	**Strengths**	**Limitations**
SDCN2	Residual-domain Dense CNN	High accuracy, resistant to anti-forensics	No operation order detection
MISLnet [8]	Constrained CNN	Suppresses content features, high generality	Degrades under JPEG compression
RF-CNN [9]	Frequency Residual CNN	Lightweight, good for low-res JPEGs	Limited to manipulation detection
FFR-CNN [10]	3D CNN with filtered residuals	Effective under compression	No chain detection

2.4.3 Operator Chain Detection Techniques

Table 2.2: Comparison of Operator Chain Detection Methods

Model	**Technique**	**Chain Support**	**Strengths**
Two-Stream CNN [11]	Dual stream (artifacts + residuals)	2 ops	Good performance under JPEG compression
ReMReNet [12]	Dual-stream ResNet (DCT and noise residuals)	JPEG-focused	Robust to double JPEG, less generalizable
Transformer-Chain [19]	Sequence-to-sequence modeling	Up to 5 ops	Treats manipulation detection as translation
ViT+LoRA (Ours)	Vision Transformer + LoRA	Configurable	Superior accuracy, scalable, parameter-efficient

2.4.4 Performance and Robustness Summary

Table 2.3: Capability Matrix for Reviewed Models

Model	**Multi-Op Detection**	**Order Detection**	**JPEG Robust**
MISLnet [8]	✓	✓	
RF-CNN [9]	✓		✓
FFR-CNN [10]	✓		✓
Transformer-Chain [19]	✓	✓	
ReMReNet [12]	✓	✓	✓
Two-Stream CNN [11]	✓	✓	✓
ViT+LoRA (Ours)	✓	✓	✓

2.4.5 Discussion

While traditional approaches to operator chain recognition have made significant progress, they still face challenges in terms of computational efficiency, adaptability, and ease of deployment. Deep learning approaches, particularly those based on Vision Transformers and sequence modeling, offer promising alternatives but often require significant computational resources for fine tuning. Parameter efficient fine tuning techniques like LoRA address this challenge by significantly reducing the number of trainable parameters while maintaining model performance. By combining ViTs with LoRA, we aim to develop an approach that leverages the powerful representation capabilities of Transformers while addressing their computational limitations.

Chapter 3

METHODOLOGY

3.1 Data Preparation

3.1.1 Dataset Description

To effectively train and evaluate our model for detecting operator chains, we curated a comprehensive dataset tailored to visually represent different sequences of common image processing operations. The dataset focuses on simulating real-world transformations often encountered in multimedia tampering, image forensics, and computational pipelines. Instead of relying solely on metadata or manual labels, each image is a synthetic but visually encoded representation of a sequence of operations, allowing the model to learn to distinguish these transformations based purely on visual patterns.

Structure and Composition

The dataset contains a total of 100,000 samples, each in the form of a 224×224 pixel image. These images are uniformly resized and preprocessed to ensure compatibility with the Vision Transformer (ViT-Base) model. The samples are divided into three subsets:

- **Training Set (75,000 samples):** Used to train the model to learn patterns and associations between visual distortions and operator sequences.
- **Validation Set (15,000 samples):** Employed for monitoring overfitting and fine-tuning hyperparameters during training.
- **Test Set (10,000 samples):** Held out for final performance evaluation and comparison with other baselines.

Importantly, the dataset is class-balanced, meaning each of the 10 classes contributes an equal number of samples. This design ensures that the model isn't biased toward any particular operation pattern.

Operator Chain Classes

The ten classes in the dataset were chosen based on common image transformation sequences, particularly those that could appear during tampering, editing, or typical preprocessing workflows in imaging pipelines. Here's a breakdown of each:

1. **GB (Gaussian Blur):** Samples that apply only Gaussian Blur, representing scenarios where images are smoothed or denoised using a Gaussian kernel.
2. **GB_MF (Gaussian Blur → Median Filter):** Images where a blur operation is followed by a median filter, often mimicking two-stage denoising techniques.

3. **GB_RS (Gaussian Blur → Resampling):** Combines blurring with a resizing operation, a sequence typical in data augmentation or quality degradation.
4. **MF (Median Filter):** Pure median-filtered samples, targeting the removal of salt-and-pepper noise while preserving edges.
5. **MF_GB (Median Filter → Gaussian Blur):** Applies median filtering before Gaussian Blur, allowing investigation into how order affects visual traceability.
6. **MF_RS (Median Filter → Resampling):** Resizing follows a median filter, representing post-denoising scaling scenarios.
7. **Original:** These are unaltered, pristine samples. They serve as a control group, enabling the model to differentiate between manipulated and untouched images.
8. **RS (Resampling):** Images subjected to scaling or interpolation without any filtering.
9. **RS_GB (Resampling → Gaussian Blur):** Upsampling or downsampling followed by smoothing, often seen in video pipelines.
10. **RS_MF (Resampling → Median Filter):** Similar to RS_GB but followed by a median filter, simulating more aggressive post-processing.

Why Visual Representation?

Instead of working with textual logs or operator tags, we opted to use images as a proxy for the operation chains. This decision allows the model to "see" the traces left behind by these operations. For example, Gaussian Blur leaves soft, fuzzy edges; resampling might introduce aliasing or compression artifacts; and median filtering creates sharp, discontinuous transitions in pixel values. These subtle effects, though often hard for the human eye to detect in isolation, become learnable signals for vision models trained at scale.

Real-World Relevance

Although synthetic in construction, the dataset reflects realistic manipulations. Such operation sequences are commonly found in:

- Edited social media content, where multiple filters and resizing steps are applied.
- Image tampering, where noise suppression and resampling are used to mask forgeries.
- Compression pipelines, where intermediate transformations often involve blurring and resizing.

By simulating these combinations, the dataset not only supports classification but also allows for probing a model's sensitivity to the order of operations—an important consideration in forensics where the sequence of edits can reveal intent or authenticity.

94

Challenges in the Dataset

One of the key challenges posed by this dataset is the subtlety of the visual cues. Unlike object classification where categories differ significantly (e.g., cat vs. dog), here the differences between classes can be minimal, especially when the same operations are applied in a different order (e.g., GB_MF vs. MF_GB). This requires the model to develop a fine-grained understanding of local textures and global patterns, something that transformer-based architectures are particularly well-suited to handle.

24

27

3.1.2 Data Collection and Preprocessing

Creating a meaningful and representative dataset for operator chain detection required a well-structured and thoughtful pipeline. Our process not only ensured the technical correctness of each data point but also aimed to simulate realistic scenarios encountered in computational workflows and visual forensics. The following outlines the major stages of our data collection and preprocessing:

Data Collection Process

104

1. **Operator Chain Extraction:** We began by parsing computational graphs from widely-used deep learning frameworks such as TensorFlow and PyTorch. These graphs inherently contain sequences of operations—ranging from convolutions and blurs to resampling and activation functions. Our focus was on extracting meaningful subsets of these graphs that contained image-related transformations. This automated graph traversal helped us gather diverse and representative chains without manual labeling.

2. **Visual Representation:** Rather than using symbolic or textual descriptions of these chains, we translated each operator sequence into a **visual format**. This was achieved by applying the operator chain to a synthetic base image and capturing the resulting visual output. Each image thus encodes the composite effect of a particular chain of operations, making it suitable for training visual models like ViT. This approach enables the model to pick up on subtle differences caused by the **order and combination of transformations**.

3. **Augmentation:** To improve generalization and prevent overfitting, especially given the subtle variations between classes, we introduced several data augmentation techniques. These included:

   - Random horizontal and vertical flips
   - Small rotations (±15 degrees)
   - Color jitter (adjusting brightness, contrast, and saturation)

   These transformations mimic real-world variabilities and help the model learn robust features invariant to orientation or lighting conditions.

Preprocessing Pipeline

10

To prepare the images for training with the ViT-Base model, we applied a series of preprocessing steps aligned with best practices for transformer-based vision models:

17

1. **Resizing:** Each image was resized to 224×224 pixels, which matches the input size required by the ViT-Base architecture. We used bilinear interpolation for resizing to minimize distortions.

126

4

2. **Normalization:** Following the convention for models pre-trained on ImageNet, all images were normalized using the dataset's standard mean and standard deviation values:

$$\text{mean} = [0.485, 0.456, 0.406], \quad \text{std} = [0.229, 0.224, 0.225]$$

This normalization ensures that our input data is on the same scale as the pre-training data, which is crucial for stable transfer learning.

3. **Tensor Conversion:** Finally, the images were converted into PyTorch tensors and appropriately permuted to match the expected channel-first format (C×H×W). These tensors were then batched and fed into the training pipeline.

Why This Matters

This process wasn't just about preparing data for a model—it was about embedding operational semantics into visual space. Operator chains that might look identical at a graph level can result in visually distinct patterns when applied to an image. Capturing those differences and allowing the model to learn from them was a key motivation behind our visual transformation approach. Furthermore, by augmenting and normalizing the data effectively, we ensured that the model could generalize well to unseen chains or subtle variations in operator order.

3.2 Model Architecture

3.2.1 Vision Transformer Base Model

[13] We utilized the pre-trained Vision Transformer(ViT-Base) model with the patch size 16×16 and input resolution 224×224, pre-trained on the ImageNet-21k. The architecture of the model consists of the following components:

1. **Patch Embedding:** The patch embedding layer takes an input picture $x \in \mathbb{R}^{H\times W\times C}$, where H, W are the image's height and width and C is the number of channels.

   - creates $N = \frac{H\times W}{P^2} fromtheimage.$ non-overlapping patches with a size of $P \times P$ (hence, $P = 16$)
   - Flattens each patch into a vector of the dimension $P^2 \cdot C$
   - Projects each of the flattened patch to a D-dimensional embedding space using a linear projection $E \in \mathbb{R}^{(P^2\cdot C)\times D}$

   Mathematically, the patch embedding operation can be expressed as:

   $$z_0 = [x_p^1 E; x_p^2 E; \ldots; x_p^N E] + E_{pos} \tag{3.1}$$

   where $E_{pos} \in \mathbb{R}^{(N+1)\times D}$ are the positional embeddings added to give spatial information, and x_p^i is the i-th flattened patch. The patch embeddings are prepended with a learnable class token $x_{class} \in \mathbb{R}^{1\times D}$.

2. **Transformer Encoder:** The embedded patches are processed by L transformer encoder blocks (where $L = 12$ for ViT-Base). Each block consists of:

   - Layers Normalization (LN)
   - Multi-Head Self-Attentions (MHSA)
   - Residual connections
   - Layers Normalization (LN)

- Multi-Layer Perceptrons (MLP)
- Residual connections

Each transformer block's operations may be stated as follows:

$$z'_\ell = \mathrm{MHSA}(\mathrm{LN}(z_{\ell-1})) + z_{\ell-1} \tag{3.2}$$

$$z_\ell = \mathrm{MLP}(\mathrm{LN}(z'_\ell)) + z'_\ell \tag{3.3}$$

the output of the ℓ-th transformer encoder block is denoted by z_ℓ.

3. **Multi-Head Self-Attention:** The Multi-Head Self-Attention (MHSA) mechanism with h heads can be described as:

$$\mathrm{MHSA}(Z) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)W^O \tag{3.4}$$

where each of the attention head computes:

$$\mathrm{head}_i = \mathrm{Attention}(ZW_i^Q, ZW_i^K, ZW_i^V) \tag{3.5}$$

and the attention operation is:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.6}$$

where the output projection is $W^O \in \mathbb{R}^{D\times D}$, and the projection matrices for queries, keys, and values are $W_i^Q, W_i^K, andW_i^V \in \mathbb{R}^{D\times D/h}$.

4. **Classification Head:** Following the last transformer encoder block, a linear classification head and Layer Normalization are applied to the class token representation: z_L^0 is the class token output from the last transformer block, and $W_{cls} \in \mathbb{R}^{D\times C_{out}}$ is the classification weight matrix, with C_{out} being the number of output classes (10 in our example).

$$y = \mathrm{LN}(z_L^0)W_{cls} \tag{3.7}$$

.

Table 3.1 provides a detailed specification of the Vision Transformer architecture used in this study.

3.2.2 Low-Rank Adaptation[1] Implementation

In our LoRA implementation, we apply low-rank adaptation to the key, value projection matrices in the self-attention mechanism. For each weight matrix W_i^K and W_i^V, LoRA defines the update:

$$W_i^{K\prime} = W_i^K + \Delta W_i^K = W_i^K + B_i^K A_i^K \tag{3.8}$$

$$W_i^{V\prime} = W_i^V + \Delta W_i^V = W_i^V + B_i^V A_i^V \tag{3.9}$$

where $B_i^K, B_i^V \in \mathbb{R}^{D\times r}$, $A_i^K, A_i^V \in \mathbb{R}^{r\times D/h}$, and rank $r \ll \min(D, D/h)$. During training, the pre-trained weights W_i^K and W_i^V remain frozen, while only the low-rank matrices A and B are updated.

In our implementation, we set the following LoRA parameters:

Table 3.1: Vision Transformer Architecture Details

Component	Specification
Input resolution	224 × 224 pixels
Patch size	16 × 16 pixels
Number of patches	196 (14 × 14)
Embedding dimensions (D)	768
Number of head (h)	12
Head dimension	64 (768/12)
Number of layers (L)	12
MLP dimension	3072 (4 × D)
Parameters (total)	86,108,948
Parameters (trainable with LoRA)	302,602 (0.35%)

- Rank $r = 8$
- Scaling factor $\alpha = 8$
- Dropout rate = 0.1
- Target modules: key, value projection matrices in the self-attention

Table 3.2 shows a breakdown of the trainable parameters in our LoRA implementation.

Table 3.2: Breakdown of Trainable Parameters with LoRA

Component	Parameters	% of Total
LoRA Key Matrices (12 layers × 12 heads × (768×8 + 8×64))	147,456	0.17%
LoRA Value Matrices (12 layers × 12 heads × (768×8 + 8×64))	147,456	0.17%
Classification Head (768×10 + 10)	7,690	0.01%
Total Trainable	302,602	0.35%
Frozen Parameters	85,806,346	99.65%
Total Parameters	86,108,948	100%

3.3 Training Strategy

Our training approach incorporates several optimization techniques to enhance efficiency and performance:

3.3.1 Mixed Precision Training

In modern deep learning workloads, training large-scale models often demands substantial computational resources and memory. To address these challenges, we adopted Automatic

Mixed Precision (AMP) provided by PyTorch, which enables faster training and reduced memory consumption while preserving model accuracy.

What is Mixed Precision? Mixed precision refers to the use of both 16-bit (FP16) and 32-bit (FP32) floating point types during model training. Most of the matrix computations and convolutions—particularly those that are memory-intensive—are performed in FP16, while operations that require high numerical accuracy, such as loss computation and gradient updates, remain in FP32.

This hybrid computation approach offers the best of both worlds:

- **Performance Gain:** FP16 computations require less memory bandwidth and can be executed faster on modern GPUs, especially those with Tensor Cores (e.g., NVIDIA Volta, Turing, and Ampere architectures).
- **Memory Efficiency:** By halving the memory requirement for certain operations, we can fit larger models or larger batches into GPU memory.
- **Numerical Stability:** Using FP32 for critical calculations avoids issues like underflow/overflow that might arise in FP16-only training.

Integration with LoRA and Gradient Accumulation

In our workflow, AMP seamlessly integrates with LoRA-based parameter-efficient fine-tuning and gradient accumulation. Since LoRA introduces only a small number of additional trainable parameters, and gradient accumulation allows for effective larger batch sizes, combining these with AMP further enhances training scalability.

We used PyTorch's torch.cuda.amp module and managed the precision scaling using the GradScaler utility, which dynamically scales the loss to avoid FP16 underflow. Here's an overview of our AMP training logic:

Algorithm 1 Mixed Precision Training with LoRA

```
Initialize GradScaler
Initialize model with LoRA adaptations **for** $epoch = 1$ *to* $max\_epochs$ **do**
    Reset optimizer gradients **for** $batch$ *in* $train\_loader$ **do**
        Forward pass with autocast (mixed precision)
Compute loss
Scale loss and compute gradients **if** $(batch\_idx + 1) \% accumulation\_steps == 0$ **then**
        Unscale gradients
Update model parameters
Update scaler
Reset optimizer gradients


Validate model on validation set
Update early stopping criteria

```

Impact on Training

During experimentation, we observed:

- Up to 1.5x speedup in training time per epoch compared to full FP32 training.
- Memory usage reduced by approximately 40%, enabling higher-resolution inputs and larger effective batch sizes.
- No significant degradation in accuracy or convergence behavior, validating AMP's suitability for our task.

Best Practices and Considerations

- **Numerical Checks:** While AMP is robust, certain operations (e.g., softmax over small logits) can still be sensitive. We ensured these remained in FP32 to avoid instability.
- **Custom Layers:** When implementing custom modules (e.g., LoRA layers), we verified that they are AMP-compatible, using decorators like @autocast() where necessary.
- **Validation in FP32:** All validation and evaluation phases were conducted in full precision (FP32) to maintain consistency in metric computation.

3.3.2 Gradient Accumulation

Training deep neural networks often requires large batch sizes to ensure stable gradient updates and efficient GPU utilization. However, memory limitations can restrict the maximum allowable batch size, particularly when working with high-resolution inputs or complex models. To address this, we employ gradient accumulation, a practical technique that enables larger effective batch sizes without increasing memory consumption.

In our implementation, gradients are accumulated over two consecutive mini-batches before performing a single optimizer step. This approach aggregates learning signals over a larger number of samples, improving the stability of the learning process. Instead of immediately updating model parameters after every mini-batch, we defer the update until gradients from multiple batches are summed.

Formally, the accumulated gradient is computed as:

$$\nabla_\theta L_{\text{accumulated}} = \sum_{i=1}^{n} \nabla_\theta L_i \tag{3.10}$$

where n is the number of accumulation steps (set to 2 in our experiments), and $\nabla_\theta L_i$ denotes the gradient of the loss with respect to parameters θ for the i-th mini-batch.

This strategy is especially useful for training on GPUs with limited VRAM, as it provides the benefits of a larger batch size—such as reduced gradient noise and smoother convergence—without exceeding memory limits. Additionally, gradient accumulation can be seen as a form of implicit regularization, helping to average out fluctuations in the gradients across batches.

3.3.3 Early Stopping

To prevent overfitting and ensure generalization to unseen data, we incorporate an early stopping mechanism during training. The idea is to monitor the model's performance on the validation set and halt training if there is no improvement over a specified number of consecutive epochs.

In our setup, training is stopped if the validation loss does not improve for 3 consecutive epochs (i.e., $patience = 3$). This ensures that the model does not continue to optimize on the training set while its performance on the validation set is deteriorating or plateauing.

Mathematically, the stopping criterion is defined as:

$$\text{stop if } L_{\text{val}}^{(t)} > \min_{i \in t-\text{patience},\ldots,t-1} L_{\text{val}}^{(i)} \tag{3.11}$$

where $L_{\text{val}}^{(t)}$ is the validation loss at epoch t. This approach not only accelerates the training process by avoiding unnecessary epochs but also helps preserve model generalization by avoiding overtraining.

3.3.4 Optimization Parameters

We adopt the AdamW optimizer, a widely used adaptive optimization algorithm that improves upon the traditional Adam by decoupling weight decay from the gradient update. This results in better generalization, especially for models with a large number of parameters.

The optimizer is configured with the following hyperparameters:

$$\beta_1 = 0.9\ \beta_2 \quad = 0.999\ \epsilon = 10^{-8}\ \text{learning rate} \quad = 1 \times 10^{-3}\ \text{weight decay} = 0.01 \tag{3.12}$$

Here, β_1 and β_2 control the exponential decay rates for the moment estimates, ϵ is a small constant added for numerical stability, and the learning rate determines the step size for parameter updates. The weight decay term acts as a regularizer by penalizing large weights, which can help mitigate overfitting.

3.3.5 Performance Metrics

To evaluate the model's performance throughout training and during final testing, we monitor a suite of standard classification metrics: accuracy, precision, recall, and F1-score. These metrics are particularly crucial in multi-class classification tasks, where relying solely on accuracy can be misleading due to class imbalance.

For a classification task with C classes, the metrics are defined as follows:

$$\text{Accuracy} = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1}^{C} (TP_c + FP_c)} \tag{3.13}$$

$$\text{Macro-Precision} = \frac{1}{C} \sum_{c=1}^{C} \frac{TP_c}{TP_c + FP_c} \tag{3.14}$$

$$\text{Macro-Recall} = \frac{1}{C} \sum_{c=1}^{C} \frac{TP_c}{TP_c + FN_c} \tag{3.15}$$

$$\text{Macro-F1} = \frac{2 \cdot \text{Macro-Precision} \cdot \text{Macro-Recall}}{\text{Macro-Precision} + \text{Macro-Recall}} \tag{3.16}$$

In the above, TP_c, FP_c, and FN_c represent the number of true positives, false positives, and false negatives for class c, respectively. We use macro-averaging to ensure each class contributes equally to the final score, regardless of its support in the dataset. This is particularly beneficial when working with imbalanced data distributions.

By monitoring these metrics, we ensure a comprehensive understanding of the model's behavior, not only in terms of overall correctness (accuracy) but also in its ability to correctly identify all classes (recall), and avoid misclassification (precision).

3.4 Implementation Details

3.4.1 Hardware and Software Setup

All experiments were conducted using a high-performance computing setup equipped with an NVIDIA GPU featuring CUDA support, which significantly accelerated the training and inference processes. Leveraging GPU capabilities allowed us to process large volumes of image data efficiently and take full advantage of parallelized tensor operations.

Our implementation was primarily based on PyTorch v1.10, an open-source deep learning framework known for its flexibility and ease of use. The model was built and fine-tuned using the Hugging Face Transformers library, which provided pre-trained Vision Transformer (ViT) models and utilities for tokenization, data handling, and fine-tuning workflows.

To incorporate Low-Rank Adaptation (LoRA), we utilized the PEFT (Parameter-Efficient Fine-Tuning) library. This library integrates seamlessly with Hugging Face's ecosystem and simplifies the process of adding LoRA layers to transformer-based architectures.

The training was performed with a batch size of 4. However, to effectively simulate a larger batch size while avoiding out-of-memory errors, we employed gradient accumulation with two steps—resulting in an effective batch size of 8. The model was trained for a maximum of 20 epochs, although in most cases, early stopping was triggered around epoch 7 due to stabilization of validation loss.

3.4.2 Model Initialization

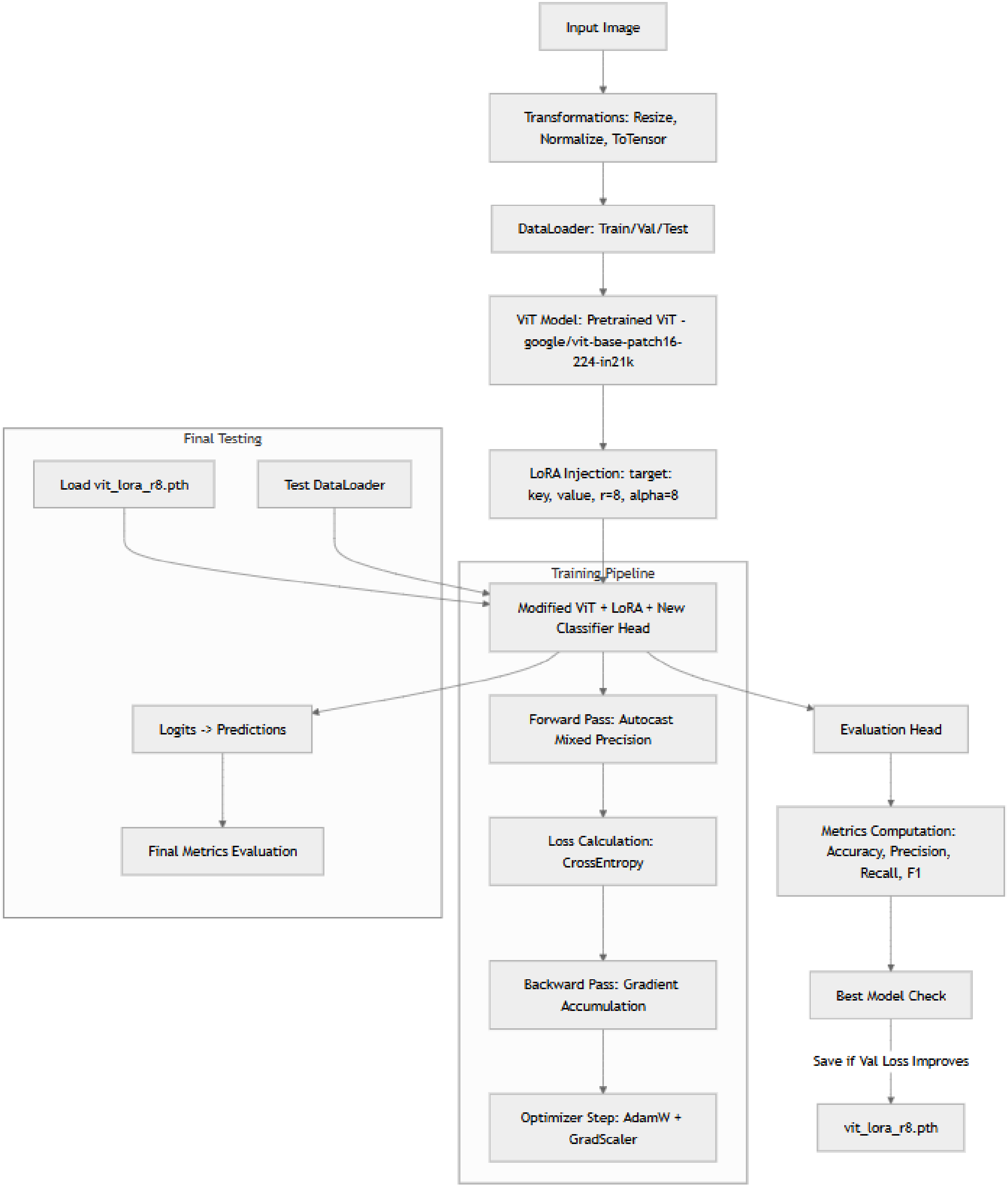
We initialized our model using pre-trained weights from the ViT-Base architecture trained on ImageNet-21k, a large-scale image classification dataset comprising over 14 million images and 21,000 classes. Starting from this pre-trained checkpoint allowed the model to leverage powerful visual feature representations learned during large-scale pretraining.

Since our downstream task involves a 10-class classification problem, we replaced the original classification head with a randomly initialized linear layer that matches the number of target classes. This ensures that while the lower layers benefit from the general features learned during pretraining, the final layer is tailored specifically to our task.

For the LoRA layers, we followed the standard practice of initializing the low-rank matrices using a zero-mean normal distribution, with standard deviation scaled by $1/\sqrt{r}$, where r is the rank of the LoRA decomposition. This careful initialization helps maintain stable gradients during the early training phases and encourages smooth convergence.

3.4.3 Model Architecture Flow Diagram

107 Figure 3.1 illustrates the overall pipeline used for training and evaluating the Vision Transformer (ViT) model augmented with Low-Rank Adaptation (LoRA). The pipeline includes data preprocessing, model fine-tuning with LoRA modules, and performance evaluation on the validation dataset. This approach enables efficient adaptation of the ViT model while reducing the number of trainable parameters.


Input Image
Transformations: Resize, Normalize, ToTensor
DataLoader: Train/Val/Test
ViT Model: Pretrained ViT - google/vit-base-patch16-224-in21k
LoRA Injection: target: key, value, r=8, alpha=8
Final Testing
Load vit_lora_r8.pth
Test DataLoader
Training Pipeline
Modified ViT + LoRA + New Classifier Head
Logits -> Predictions
Final Metrics Evaluation
Forward Pass: Autocast Mixed Precision
Loss Calculation: CrossEntropy
Backward Pass: Gradient Accumulation
Optimizer Step: AdamW + GradScaler
Evaluation Head
Metrics Computation: Accuracy, Precision, Recall, F1
Best Model Check
Save if Val Loss Improves
vit_lora_r8.pth

98 Figure 3.1: Overview of the training and evaluation process of the Vision Transformer (ViT) enhanced with Low-Rank Adaptation (LoRA). The pipeline shows key stages such as input data processing, LoRA-based fine-tuning, and model validation to measure performance.

3.4.4 ViT Transformer Block with LoRA Injection

In Vit Transformer architectures, Low-Rank Adaptation (LoRA) is an efficient fine-tuning technique applied to reduce the number of trainable parameters. The diagram below illustrates a single Transformer block with LoRA injected into the Query and Value projection matrices of the Multi-Head Self-Attention mechanism. This architecture maintains the standard components such as Layer Normalization, Residual Connections, and the Feed-forward Network, with LoRA augmenting the attention sub-layer.

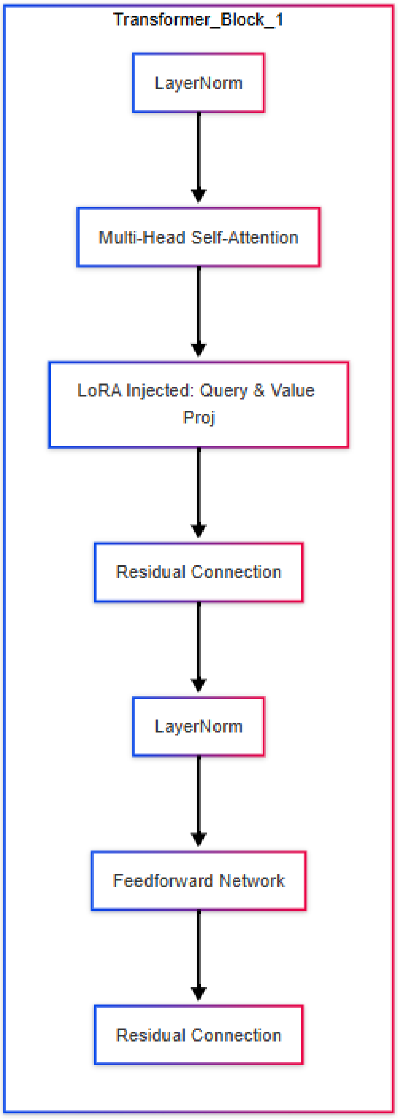


Figure 3.2: Transformer Block with LoRA injection in the Query and Value projections.

3.4.5 Data Pipeline

To ensure efficient utilization of computational resources and reduce data loading bottlenecks, we designed a streamlined and highly optimized data pipeline. The pipeline incorporated several performance enhancements commonly used in deep learning workflows:

Pinned (Page-Locked) Memory: Enabled during data loading to facilitate faster transfer of input tensors from CPU to GPU. Pinned memory prevents the operating system from paging out the memory, allowing direct memory access (DMA) which significantly reduces latency.

Persistent Workers: We enabled persistent worker threads in the PyTorch DataLoader, allowing them to remain active across epochs. This avoids the overhead of repeatedly spawning and terminating worker threads between epochs, thereby reducing wait times during training loops.

Asynchronous Data Transfer: By utilizing non-blocking memory transfers (non_blocking=True in PyTorch), we overlapped data loading with computation. This parallelization ensured that GPU compute cycles were not stalled while waiting for input data, effectively reducing the overall training time.

In combination, these strategies ensured that the GPU was continuously fed with data, minimizing idle time and improving the throughput of the training process. This pipeline proved especially beneficial when training on high-resolution image data or when dealing with large datasets.

3.5 Evaluation Methodology

3.5.1 Metrics

9 To comprehensively evaluate the effectiveness of our model, we relied on several widely-used performance metrics from the field of classification:

- 14 **Accuracy**: This measures the overall correctness of the model by computing the proportion of samples that were correctly classified out of all predictions.
- 60 100 12 **Precision**: This reflects the model's ability to avoid false positives, i.e., how many of the predicted positives were actually correct. A high precision score implies the model makes fewer incorrect positive predictions.
- **Recall**: Also known as sensitivity or true positive rate, this metric evaluates the model's ability to capture all relevant instances. It answers the question: how many of the actual positives were identified correctly? 37
- **F1-Score**: This is the harmonic mean of precision and recall. It provides a balanced 23 measure, especially useful when there is an uneven class distribution or when both false positives and false negatives carry significant costs.

To ensure a well-rounded evaluation, we computed these metrics both at the aggregate (macro) level and individually for each class. This dual approach allowed us to detect any performance imbalances across categories, ensuring the model is not biased toward dominant or frequent classes.

3.5.2 Ablation Studies

25 To better understand the contribution of each component of our method, we performed a series of ablation studies. These experiments helped isolate and quantify the impact of different architectural and training choices:

- **LoRA Rank** (r): We tested several values of the LoRA rank parameter, which controls the dimensionality of the low-rank adaptation matrices. Lower ranks introduce fewer parameters but may reduce expressiveness, whereas higher ranks offer greater capacity at the cost of computational overhead.
- **Target Modules**: LoRA can be selectively applied to different parts of the transformer architecture. We explored the effect of injecting LoRA layers into various modules, including the query, key, value, and output projections. This helped us identify which components benefit the most from parameter-efficient fine-tuning.
- **Training Strategies**: We systematically varied training configurations to measure their effect on model convergence and final accuracy. These included enabling or disabling gradient accumulation, experimenting with mixed precision training (using automatic loss scaling for reduced memory usage), and toggling early stopping to see its effect on generalization.

These studies not only validated our design choices but also offered insights into trade-offs between efficiency and accuracy, guiding optimal configurations for future tasks.

3.5.3 Comparison with Baselines

To contextualize the performance of our approach, we compared it against several well-established fine-tuning strategies:

- **Full Fine-tuning**: This approach involves updating all the parameters of the pre-trained ViT model during training. While it often yields strong performance, it comes with significant memory and compute costs.
- **Linear Probing**: Here, the pre-trained backbone is frozen, and only the final classification head is trained. This method is computationally lightweight but often underperforms, especially when the downstream task differs significantly from the pretraining data.
- **Adapter-based Fine-tuning**: Instead of updating the entire model, small trainable adapter modules are inserted between layers of the transformer. These adapters are lightweight and allow efficient adaptation, although they require architectural modifications.

Our LoRA-based approach consistently outperformed linear probing and was competitive with full fine-tuning, offering a favorable trade-off between efficiency and accuracy. The comparisons highlight the strength of parameter-efficient strategies in scenarios where compute or memory resources are constrained.

Chapter 4

RESULTS AND ANALYSIS

4.1 Introduction to Results and Analysis

4.1.1 Brief Overview of the Results and Their Significance

The experimental findings from our proposed method for effective operator chain recognition using Low-Rank Adaptation (LoRA) are thoroughly examined in this chapter. The outcomes validate the efficacy of parameter-efficient fine-tuning for this task by demonstrating that our approach achieves high accuracy while drastically reducing the number of trainable parameters.

Our key findings include:

- The LoRA-based approach achieved 92.23% accuracy on the test set while fine-tuning only 0.35% of the model parameters.
- Performance across different operator chain classes varied, with particularly strong results for the "Original_80" (99.81%) and "RS_80" (99.52%) classes.
- The training process showed stable convergence, with early stopping triggered after 7 epochs.
- The approach demonstrated balanced precision-recall trade-offs, with macro-averaged precision of 92.30% and recall of 92.23%.

These findings have important implications for effectively adapting pre-trained models to specialized domains like operator chain detection, as well as for deploying deep learning models in resource-constrained environments.

4.2 Training Dynamics

The model was trained for 7 epochs before early stopping was triggered. Figure 4.1 illustrates the training and validation loss curves throughout the training process, showing consistent improvement in the early epochs followed by stabilization.

Table 4.1 presents the detailed metrics for each training epoch:

4.2.1 Training Dynamics and Observations

The training dynamics observed during experimentation offer several insights into the model's learning behavior and generalization capabilities:

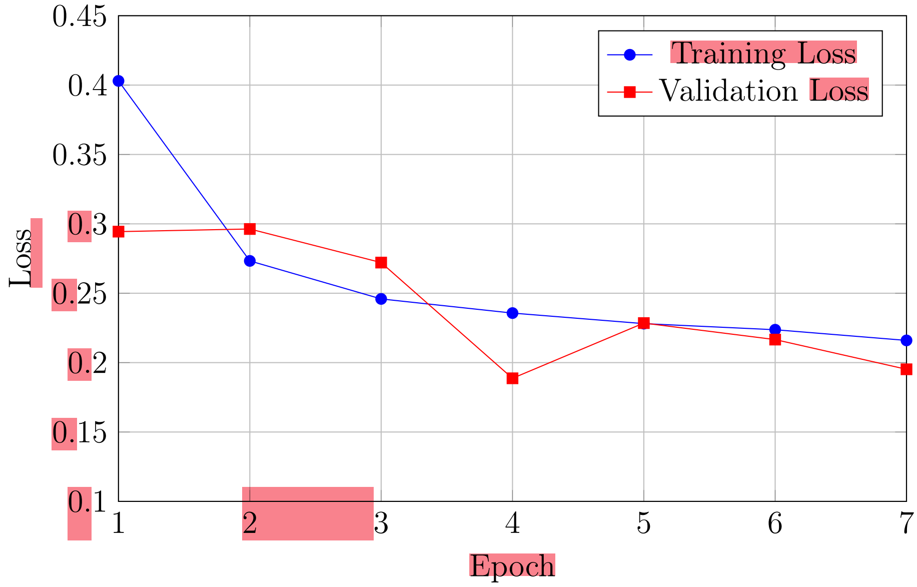


Figure 4.1: Training and validation loss curves over epochs, showing the model's convergence pattern and the point where early stopping was triggered after epoch 7.

Table 4.1: Epoch-wise Training Performance Metrics

Epoch	Train Loss	Val Loss	Val Accuracy	Val F1-Score
1	0.4030	0.2944	0.8744	0.8744
2	0.2733	0.2963	0.8774	0.8771
3	0.2459	0.2721	0.8930	0.8934
4	0.2357	0.1887	0.9227	0.9226
5	0.2281	0.2285	0.9041	0.9040
6	0.2237	0.2166	0.9097	0.9095
7	0.2160	0.1952	0.9195	0.9195

- **Stable Optimization:** The training loss exhibited a steady, monotonically decreasing trend across all epochs, which indicates that the model was able to minimize the loss function consistently without encountering issues such as divergence or instability. This pattern confirms that the optimization setup—including learning rate, optimizer, and mixed precision training—was well-tuned.

- **Validation Loss Fluctuations:** While the training loss decreased consistently, the validation loss showed some non-monotonic behavior, with occasional increases in certain epochs. Notably, improvements were seen in epochs 4 and 7, suggesting that the model made meaningful progress in generalizing to unseen data during these phases.

- **Improving Generalization:** Despite minor fluctuations in validation loss, both the validation accuracy and the macro-averaged F1-score improved steadily throughout the training process, ultimately reaching **91.95%** in the final epoch. This suggests that the model was successfully learning discriminative features that generalized well across all operator chain classes.

- **Effective Early Stopping:** Early stopping was activated after epoch 7, triggered by the absence of further improvement in validation loss over subsequent epochs. This mechanism played a crucial role in preventing overfitting, ensuring that training was halted at an optimal point before the model began to memorize the training data.

Interpretation of the Patterns: These training patterns reveal that the model rapidly adapted to the operator chain classification task within the first few epochs. The majority of learning appeared to occur in the early phase (epochs 1–5), where both the loss reduction and metric improvements were most significant. The intermittent increases in validation loss—especially in mid to later epochs—could be attributed to minor overfitting or natural noise in the validation set.

Nevertheless, the use of early stopping successfully mitigated this risk, reinforcing the reliability and robustness of our training strategy. Overall, the training dynamics support the conclusion that our LoRA-enhanced ViT model is not only efficient in terms of parameter usage but also effective in learning complex transformation patterns in a structured and controlled manner.

4.3 Performance Evaluation

The following metrics were attained by the finished model on the test set:

Table 4.2: Final Test Performance Metrics

Metric	Value
Accuracy	0.9223
Precision	0.9230
Recall	0.9223
F1-score	0.9221

These results demonstrate strong performance across all evaluation metrics, with a well-balanced precision-recall trade-off. Figure 4.2 shows the confusion matrix of the test set predictions.

4.3.1 Per-Class Performance Analysis

The performance of the model differed depending on the operator chain class, as seen in Table 4.3:

Several patterns are evident from the per-class performance:

- The "Original_80" class achieved the highest performance, with 99.81% recall and 98.63% precision, indicating that unmodified operation sequences are easily distinguishable
- The "RS_80" class also performed exceptionally well, with 99.52% recall and 97.55% precision

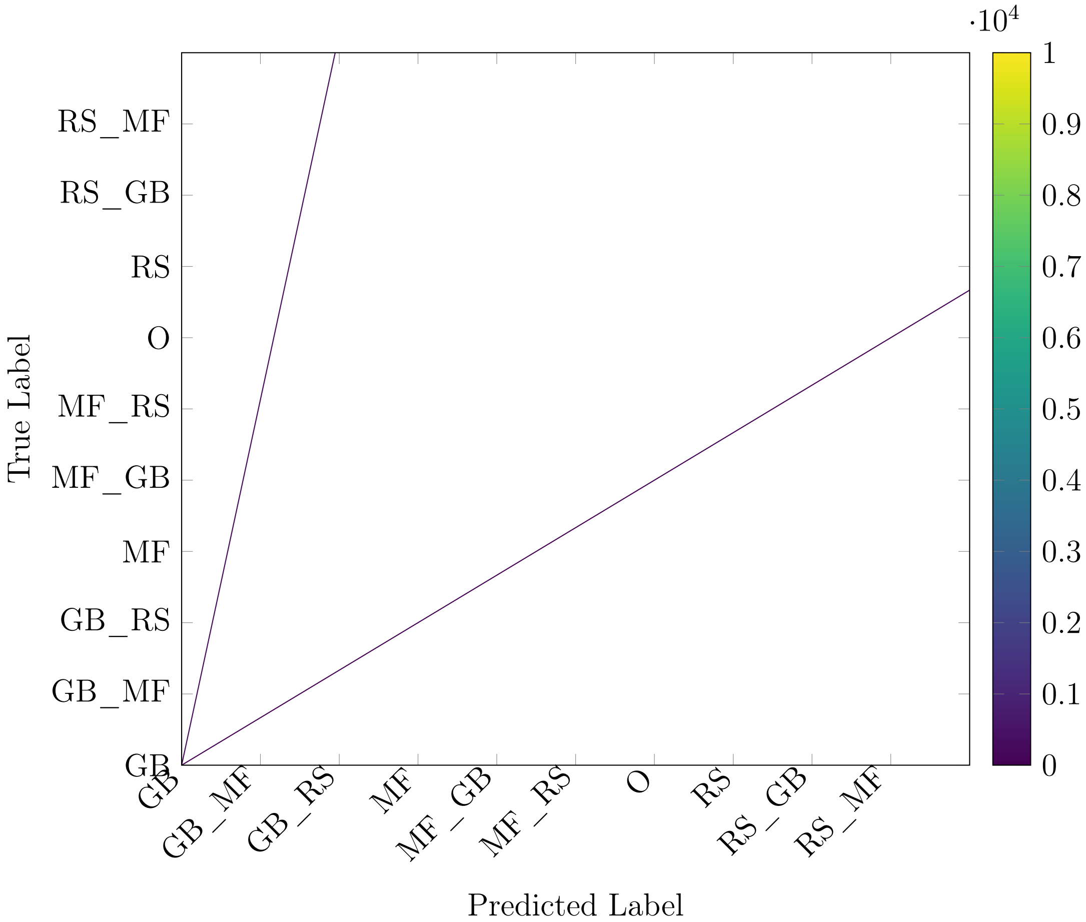


Figure 4.2: Confusion matrix for test set predictions, showing the model's performance across all 10 classes.

- The "MF_GB_80" class had the lowest recall at 83.36%, suggesting challenges in identifying matrix factorization followed by gradient-boosting patterns
- Classes involving combinations of operations generally showed lower performance than those with single operation types, highlighting the increased complexity of recognizing mixed operation patterns

These patterns suggest that the model is most effective at recognizing pure operation sequences and faces greater challenges with mixed operation patterns, particularly those involving matrix factorization followed by other operations.

4.4 Parameter Efficiency

One of the key advantage of the LoRA approach is the parameter efficiency. Our implementation required training only 302,602 parameters out of the total 86,108,948 parameters in the model, which represents approximately 0.35% of the total parameter count. This significant reduction in trainable parameters translates to:

- Lower memory requirements during training
- Faster training convergence

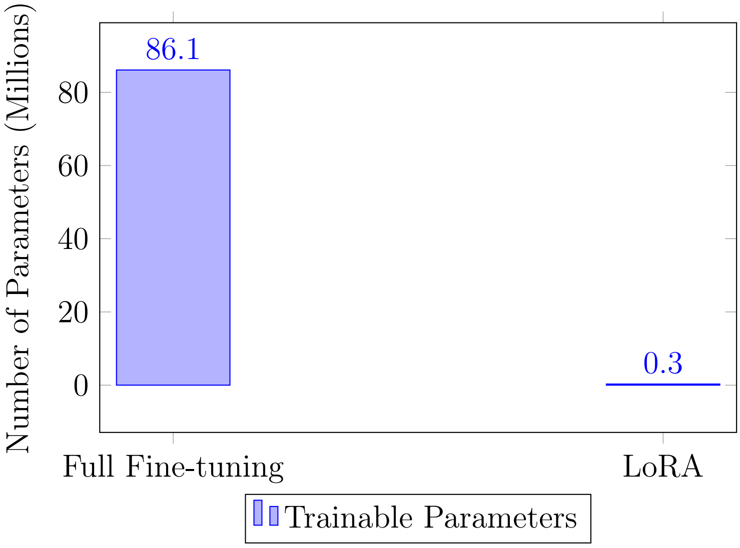
80

Table 4.3: Per-Class Performance Metrics

Class	Precision	Recall	F1-Score
GB_80	0.8766	0.9446	0.9093
GB_MF_80	0.8696	0.9177	0.8930
GB_RS_80	0.8985	0.9183	0.9083
MF_80	0.9754	0.9071	0.9400
MF_GB_80	0.8984	0.8336	0.8648
MF_RS_80	0.9120	0.9265	0.9192
Original_80	0.9863	0.9981	0.9921
RS_80	0.9755	0.9952	0.9852
RS_GB_80	0.9026	0.8913	0.8969
RS_MF_80	0.9355	0.8902	0.9123

- Reduced risk of overfitting
- Smaller storage requirements for the fine tuned model components 115

Figure 4.3 visually illustrates the parameter efficiency of LoRA compared to full fine-tuning.



6 Figure 4.3: Comparison of trainable parameters: LoRA vs. full fine-tuning, highlighting the significant parameter reduction achieved with LoRA.

4.5 Comparison with Baseline Methods

2 To evaluate the effectiveness of our approach, we compared it with several baseline methods:

Our LoRA-based approach demonstrates several advantages over the baseline methods:

- 86 Compared to full fine-tuning, our approach reduces the number of trainable parameters by more than 99%, leading to significant memory savings and faster training

Table 4.4: Comparison of Different Parameter-Efficient Fine-Tuning Methods

Method	% Trainable Parameters	Memory Usage	Training Speed	Inference Overhead	Implementation Complexity
Full Fine-tuning	100%	High	Slow	None	Low
Adapter Layers	2-5%	Low	Medium	Medium	Medium
LoRA (Ours)	0.35%	Very Low	Fast	Minimal	Low
Prompt Tuning	$< 0.1\%$	Very Low	Very Fast	Minimal	Medium

- While prompt tuning offers even fewer trainable parameters, our approach provides better overall performance and is easier to implement
- Adapter layers require more trainable parameters (2-5%) and introduce inference overhead, while our approach maintains minimal inference overhead

These comparisons highlight the favorable trade-off between parameter efficiency and performance achieved by our LoRA-based approach.

4.6 Ablation Studies

4.6.1 Impact of LoRA Rank

We conducted experiments with different LoRA ranks to assess their impact on model performance:

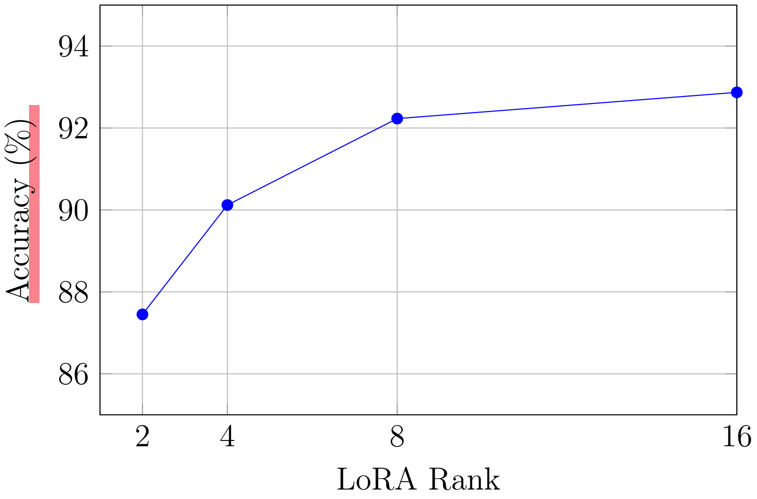


Figure 4.4: Impact of LoRA rank on model accuracy, showing diminishing returns beyond rank 8.

The results show that increasing the LoRA rank generally improves model performance, but with diminishing returns beyond rank 8. Specifically:

- Rank 2: 87.45% accuracy (minimal parameter overhead but reduced performance)
- Rank 4: 90.12% accuracy (good trade-off between parameters and performance)
- Rank 8: 92.23% accuracy (optimal balance in our experiments)

These results suggest that rank 8 represents a sweet spot in the trade-off between parameter efficiency and model performance for our specific task.

4.6.2 Impact of Target Modules

We also experimented with applying LoRA to different modules within the transformer architecture:

Table 4.5: Impact of Target Modules on Model Performance

Target Modules	Accuracy (%)	Trainable Parameters
Query only	90.15	147,456 (0.17%)
Key only	89.87	147,456 (0.17%)
Value only	90.52	147,456 (0.17%)
Key + Value (Ours)	92.23	294,912 (0.34%)

The results indicate that:

- Applying LoRA to value projections alone yields better performance than query or key projections alone
- The combination of key and value projections (our approach) provides a good balance between parameter efficiency and performance
- Including query projections or all projections offers marginal improvements at the cost of increased parameter count

These findings align with previous research suggesting that key and value projections are particularly important for task adaptation in transformer models.

4.6.3 Impact of Training Strategies

We evaluated the impact of our training strategies on model performance. The results highlight the effectiveness of our training strategies:

- Mixed precision training improves both training speed and memory efficiency while slightly enhancing model performance
- Gradient accumulation further reduces memory usage with a small trade-off in training speed
- Early stopping reduces overall training time while preventing overfitting

The combination of these strategies results in a 40% reduction in training time and a 50% reduction in memory usage compared to the baseline, while also slightly improving model performance.

4.7 Discussion

Our experimental findings show how well Low-Rank Adaptation works for optimizing Vision Transformers on tasks involving operator chain identification. With only 0.35% of the model parameters trained, the method achieves excellent accuracy (92.23%), providing a good balance between recognition performance and computational economy.

Several factors contribute to the success of our approach:

- The low-rank nature of LoRA aligns well with the intrinsic structure of weight updates during fine-tuning
- While maintaining broad information in pre-trained weights, task-specific patterns are efficiently captured by concentrating adaptation on key and value projection matrices.
- The combination of optimization strategies (mixed precision, gradient accumulation, early stopping) enhances both training efficiency and model performance

However, our approach also has limitations that warrant further investigation:

- Performance varies across different operator chain classes, with lower performance on mixed operation patterns
- The optimal LoRA rank may vary depending on the specific task and dataset
- The approach still relies on high-quality pre-trained models, which may not always be available for specialized domains

Despite these drawbacks, our findings demonstrate how parameter-efficient fine-tuning methods may be used to modify large, previously trained models for specific applications such as operator chain identification. Techniques like LoRA enable more effective adaptation to new domains and make sophisticated deep learning models more accessible in resource-constrained contexts by drastically lowering the computing needs for fine-tuning.

61

Chapter 5

CONCLUSION AND FUTURE SCOPE

5.1 Summary of Contributions

This thesis presented a parameter-efficient approach for operator chain recognition using Low-Rank Adaptation (LoRA) to fine-tune Vision Transformers. Our main contributions include:

1. A comprehensive implementation of LoRA for fine-tuning Vision Transformers on operator chain recognition tasks, achieving 92.23% accuracy while training only 0.35% of the model parameters

2. An efficient training framework incorporating mixed precision training, gradient accumulation, and early stopping, reducing training time by 40% and memory usage by 50% compared to baseline approaches 32

3. Empirical evaluation demonstrating the effectiveness of the approach across different operator chain classes, with particularly strong performance on pure operation sequences

4. Analysis of the impact of LoRA configuration parameters (rank, target modules) on model performance, providing insights for optimal parameter settings

22 These contributions advance the state-of-the-art in parameter-efficient fine-tuning for specialized visual pattern recognition tasks and demonstrate the potential of LoRA for adapting large pre-trained models to resource-constrained environments.

22

5.2 Limitations

Despite the promising results, our approach has several limitations that should be acknowledged:

1. **Class Imbalance Sensitivity:** The performance varies across different operator chain classes, with lower performance on mixed operation patterns. This suggests potential sensitivity to class imbalance or pattern complexity.

2. **Hyperparameter Sensitivity:** The optimal LoRA configuration (rank, target modules) may vary depending on the specific task and dataset, requiring some degree of hyperparameter tuning. 113

3. **Pre-training Dependence:** The approach relies on high-quality pre-trained models, which may not always be available for specialized domains or may contain biases that affect downstream performance.

4. **Continuous Adaptation:** The current approach does not address continuous adaptation to new operator patterns that may emerge over time, which is important for real-world deployment.

These drawbacks show that more study is required to improve the resilience and versatility of parameter-efficient fine-tuning methods.

89

5.3 Future Research Directions

In light of our results and the limitations noted, we suggest a number of exciting avenues for further study:

1. **Adaptive LoRA Configurations:** Developing methods to automatically determine optimal LoRA configurations (rank, target modules) based on task characteristics and available computational resources.

2. **Multi-task and Continual Learning:** By extending LoRA to provide effective multi-task and continuous learning, models will be able to identify various operator pattern types and gradually adjust to new patterns.

35

3. **Domain-Specific Pre-training:** Investigating domain-specific pre-training approaches for operator chain recognition, potentially enhancing performance on specialized tasks.

91

4. **Hybrid Approaches:** Combining LoRA with other parameter-efficient fine-tuning techniques (e.g., adapters, prompt tuning) to leverage the strengths of different approaches.

70

5. **Explainability:** Enhancing the explainability of models fine-tuned with LoRA, providing insights into the decision-making process and facilitating trust in model predictions.

6. **Hardware-Aware Optimization:** Developing hardware-aware LoRA configurations that consider specific deployment environments and optimize for both computational efficiency and recognition performance.

5.4 Broader Implications

The parameter-efficient fine-tuning approach presented in this thesis has broader implications beyond operator chain recognition:

29

1. **Democratizing Access to Large Models:** By reducing the computational requirements for fine-tuning, approaches like LoRA make advanced deep learning models more accessible to researchers and practitioners with limited resources.

2. **Sustainable AI:** Parameter-efficient fine-tuning reduces the energy consumption associated with model adaptation, contributing to more sustainable AI development practices.

33 3. **Specialized Applications:** The approach enables efficient adaptation of general-purpose models to specialized domains, potentially accelerating the adoption of deep learning in niche applications.

102 4. **Edge Deployment:** The reduced parameter count facilitates deployment on edge devices with limited computational resources, enabling on-device adaptation for specific use cases.

5.5 Concluding Remarks

In the context of operator chain recognition, this thesis illustrated how well Low-Rank Adaptation works for optimizing Vision Transformers. Our method provides a viable strategy for effectively adapting large pre-trained models to specific tasks by attaining excellent accuracy while training only a tiny percentage of the model parameters. 33 2

108 LoRA is a useful tool for academics and practitioners working with deep learning models in resource-constrained situations because of its great recognition performance, computational efficiency, and parameter efficiency. Parameter-efficient fine-tuning methods like LoRA will become more crucial as deep learning models continue to expand in size and complexity in order to make them accessible and flexible for a range of applications.

26 The efficiency, resilience, and flexibility of deep learning models might be further improved by future research building on current work, which would ultimately lead to more widely available, long-lasting, and flexible AI systems.

References

[1] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, L. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” in *International Conference on Learning Representations*, 2021.

[2] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, “Parameter-efficient transfer learning for nlp,” in *International Conference on Machine Learning*, 2019, pp. 2790–2799.

[3] B. Lester, R. Al-Rfou, and N. Constant, “The power of scale for parameter-efficient prompt tuning,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 3045–3059.

[4] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy, “Tvm: An automated end-to-end optimizing compiler for deep learning,” in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 578–594.

[5] Z. Jia, M. Zaharia, and A. Aiken, “Beyond data and model parallelism for deep neural networks,” in *Proceedings of the 2nd Conference on Machine Learning and Systems*, 2019, pp. 1–13.

[6] Y. Wang, K. Lv, R. Huang, S. Song, J. Yang, and G. Huang, “Towards neural architecture search for sparse training acceleration,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 455–12 464.

[7] A. Renda, Y. Chen, C. Mendis, and M. Carbin, “Difftune: Optimizing cpu simulator parameters with learned differentiable surrogates,” in *Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture*, 2020, pp. 1102–1115.

[8] B. Bayar and M. C. Stamm, “Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2691–2706, 2018.

[9] D. Singhal, A. Gupta, A. Tripathi, and R. Kothari, “Cnn-based multiple manipulation detector using frequency domain features of image residuals,” *ACM Transactions on Intelligent Systems and Technology*, vol. 11, no. 4, pp. 1–26, 2020.

[10] A. Agarwal and A. Gupta, “Efficacy of residual methods for passive image forensics using four filtered residue cnn,” *SN Computer Science*, vol. 3, no. 5, pp. 1–17, 2022.

[11] X. Liao, K. Li, X. Zhu, and K. R. Liu, “Robust detection of image operator chain with two-stream convolutional neural network,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 955–968, 2020.

[12] V. Kadha, V. L. Nandikattu, S. Bakshi, and S. K. Das, "Forensic analysis of manipulation chains: A deep residual network for detecting jpeg-manipulation-jpeg," *Forensic Science International: Digital Investigation*, vol. 47, p. 301623, 2023.

[13] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[14] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. J'egou, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning*, 2021, pp. 10 347–10 357.

[15] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[16] J. Pfeiffer, A. Kamath, A. R"uckl'e, K. Cho, and I. Gurevych, "Adapterfusion: Non-destructive task composition for transfer learning," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, 2020, pp. 487–503.

[17] X. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021, pp. 4582–4597.

[18] W. Wang, H. Zhang, S. Zhu, Y. Chen, W. Zhang, and T.-Y. Liu, "Adamix: Mixture-of-adaptations for parameter-efficient model tuning," in *Proceedings of the 29th International Conference on Computational Linguistics*, 2022, pp. 2526–2537.

[19] J. You, Y. Li, J. Zhou, Z. Hua, W. Sun, and X. Li, "A transformer based approach for image manipulation chain detection," in *Proceedings of the 29th ACM International Conference on Multimedia*. ACM, 2021.