**Project Report**

***Enhancing Machine Learning Development Efficiency through DevOps and MLOps***

**Master’s in computer science**

<https://github.com/RajeshRamadas/Yolo8-Annotation-Tool.git>

Student: Rajesh Kumar Ramadas

rajesh-kumar.ramadas@iu-study.org

Matriculation: 92125100

Course name                      Project: Master Thesis

(MMTHE01)

University: International University of Applied Sciences

Juri-Gagarin-Ring 152 · D-99084 Erfurt

University Supervisor: Dr. Aditya Mushyam

aditya.mushyam@iu.org

Submission Date:  April 2025

Contents

[1. Introduction 2](#_Toc194317870)

[2. System Architecture 2](#_Toc194317871)

[3. Features 2](#_Toc194317872)

[3.1. Neural Architecture Search 2](#_Toc194317873)

[3.2. Model Architecture Generation 2](#_Toc194317874)

[3.3. Data Augmentation 3](#_Toc194317875)

[3.4. Configuration Management 3](#_Toc194317876)

[3.5. Pruning & Quantization 3](#_Toc194317877)

[3.6. Knowledge Distillation 3](#_Toc194317878)

[3.7. Jenkins (CI/CD) 3](#_Toc194317879)

[8.9. Application on Raspberry pi 3](#_Toc194317880)

[4. Technological stack 3](#_Toc194317881)

[5. Implementation Details 3](#_Toc194317882)

[6. Workflow and Usage 3](#_Toc194317883)

[7. Limitations 3](#_Toc194317884)

[8. Repository & Prerequisites 3](#_Toc194317885)

[9. Performance Analysis 3](#_Toc194317886)

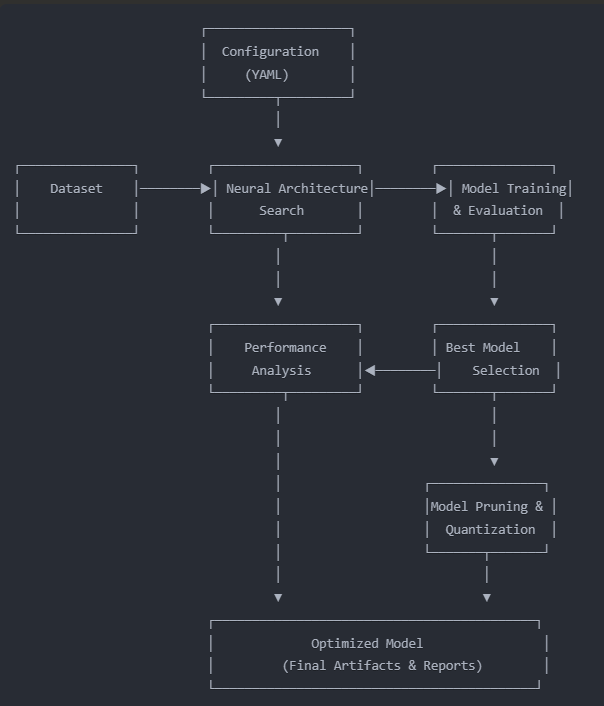
[11. Conclusions and Future Work 3](#_Toc194317887)

[12. Bibliography 3](#_Toc194317888)

## 1. Introduction

## 2. System Architecture

The framework follows a modular design pattern, with distinct components for architecture search, model training, evaluation, and optimization. The high-level architecture is illustrated below:



The modular design allows for flexibility and extensibility, enabling users to:

* Use the entire pipeline for end-to-end optimization
* Focus exclusively on architecture search
* Apply only pruning and quantization to existing models
* Best architecture is used for knowledge distillation
* Trained model is deployed for application

This architecture is implemented through several Python modules that handle specific aspects of the optimization process.

## 3. Features

## 3.1. Neural Architecture Search

## 3.2. Model Architecture Generation

## 3.3. Data Augmentation

## 3.4. Configuration Management

## 3.5. Pruning & Quantization

## 3.6. Knowledge Distillation

## 3.7. Jenkins (CI/CD)

## 8.9. Application on Raspberry pi

## 4. Technological stack

## 5. Implementation Details

## 6. Workflow and Usage

## 7. Limitations

## 8. Repository & Prerequisites

## 9. Performance Analysis

## 11. Conclusions and Future Work

## 12. Bibliography

*opencv*. (n.d.). Retrieved from https://opencv.org/: https://opencv.org/

*python* . (n.d.). Retrieved from https://www.python.org/: https://www.python.org/

*python-pillow*. (n.d.). Retrieved from https://python-pillow.org/: https://python-pillow.org/

*ultralytics* . (n.d.). Retrieved from https://pypi.org/project/: https://pypi.org/project/ultralytics/