Machine Learning Internship

Manual 01

Prepared by
ARCH TECHNOLOGIES

Email: archtechnologies.pk@gmail.com
LinkedIn: www.linkedin.com/company/archtechnologiespk/

Contents

1 Copyright Notice **1**

2 About the Company **2**

3 Introduction **3**

4 Reference Book **4**

5 Category Selection **5**
5.1 Guidelines for Selecting a Category 5

6 Category A: Advanced **6**
6.1 Task 01 6
6.2 Task 02 8

7 Category B: Intermediate **10**
7.1 Task 01 10
7.2 Task 02 12

8 Category C: Beginner **13**
8.1 Task 01 13
8.2 Task 02 14

9 Supplementary Tasks (Optional) **15**
9.1 Installing DeepSeek on Your PC 15
9.2 Creating Virtual Machines and Installing Ubuntu Linux 16

10 Submission Guidelines **17**
10.1 What to Include in Your Report? 17
10.2 Additional Content 17
10.3 File Naming Format 18
10.4 Submission Deadline 18

11 Evaluation Criteria **19**

12 Acknowledgements **20**

1. Copyright Notice

2. About the Company

ARCH Technologies is an initiative dedicated to promoting and raising awareness about the urgent need for the development and adoption of artificial intelligence across the country. Started with the vision of empowering local talent and industries through cutting-edge AI research, education, and innovation, it aims to bridge the gap between emerging global advancements and our national capabilities. We strive to build a sustainable AI ecosystem that drives progress, solves real-world problems, and ensures our nation remains competitive in the digital future.

3. Introduction

Welcome to the ARCH Technologies **Internship Manual 01**. This manual outlines the procedures, tasks, and submission guidelines for our Machine Learning Internship Students. Our aim is to promote consistency, accountability, and productivity, ensuring that all team members have clear expectations and resources.

While we encourage the use of technology to enhance learning, the use of Artificial Intelligence (AI) tools to generate or complete tasks is strictly prohibited. All work must reflect your own understanding and effort. However, teamwork and collaboration are highly encouraged. We advise members to study in groups, discuss concepts, and support one another — as collective learning often leads to deeper insights and stronger problem-solving skills.

To stay on track, we recommend dedicating at least **two hours a day** to your tasks and studies. Consistent effort not only builds technical skills but also strengthens your ability to tackle real-world AI challenges.

By following these guidelines, you'll cultivate both individual expertise and a collaborative mindset, aligning with our mission to advance AI and ML innovation.

Note

This manual contains clickable links for your convenience. Simply click on the blue text to visit the relevant pages, websites or watch tutorial videos. If the links do not work, please try opening the PDF file in Google Chrome. You can also copy and paste the Name (of file/video) or URL directly into your browser or YouTube search bar.

4. Reference Book

We will be following the book **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow** by **Aurélien Géron** as our primary resource for theory and practicals.

This book is chosen because it provides a balanced approach to both theoretical concepts and practical implementations. It emphasizes hands-on learning through coding exercises and real-world examples, helping you not only understand the underlying principles of Machine Learning but also apply them effectively.

Following a structured book ensures that you learn concepts in a logical sequence, build a strong foundation, and avoid the confusion that can come from scattered online resources. It also encourages consistency and depth in your study.

We highly recommend dedicating time to both the book's theory sections and its practical exercises to get the most out of this learning experience.

For access to the book, you can visit the following website:

- Hands–On Machine Learning with Scikit–Learn and TensorFlow 2e PDF Drive Link

If you cannot download the book, feel free to contact us by email and we will provide it to you.

5. Category Selection

We have divided tasks into three categories based on experience level and knowledge :

1. **Category A: Advanced Tasks** — For students who have fairly good knowledge of Machine Learning and are looking to advance their concepts and experience.
2. **Category B: Intermediate Tasks** — Suitable for those students who have some knowledge of ML and Python or any other programming language in general.
3. **Category C: Beginner Tasks** — Intended for those students who do not have any prior coding experience in any programming language and are new to this field.

Please select the category that best matches your current expertise. If you feel you've chosen the wrong category, don't worry — you are free to change your category at any time. We encourage you to be sincere to yourself; if you find the tasks too easy, consider moving to a higher category for challenging tasks.

5.1 Guidelines for Selecting a Category

To help you choose the most suitable category, we provide the following recommendations based on your academic background:

- **Category A: Advanced Tasks** — Recommended for students enrolled in degree programs related to Artificial Intelligence, Data Science, or closely aligned fields.
- **Category B: Intermediate Tasks** — Suitable for students pursuing degrees in Computer Science, Software Engineering, Computer Engineering, or similar disciplines.
- **Category C: Beginner Tasks** — Suggested for students from Mathematics, Physics, Statistics, Law, or other non-computing fields who are new to programming and AI concepts.

These guidelines are meant to assist you in making an informed choice. However, **you are free to select any category** based on your confidence and experience level. If you find the tasks too simple or too challenging, you can always switch to a more appropriate category as you progress.

6. Category A: Advanced

6.1 Task 01

Time Series Analysis of Ethereum (ETH/USDT) Market Projections using ARIMA

Objective: The objective of this task is to analyze historical Ethereum (ETH/USDT) price data using Autoregressive Integrated Moving Average (ARIMA) models. Time series analysis focuses on understanding data points ordered in time, identifying underlying patterns such as trends, seasonality, and cyclic behavior. ARIMA models combine autoregression (AR), differencing (I), and moving averages (MA) to capture these patterns and make reliable forecasts. The goal is to leverage this method to predict future prices and volatility in Ethereum's market, providing data-driven insights that support strategic decision-making and enhance our understanding of cryptocurrency market dynamics.

Task Requirements:

1. **Data Collection & Preparation:** Source reliable historical price data for Ethereum (ETH/USDT). Suggested platforms include:

   - CoinGecko (API for historical cryptocurrency data)
   - Binance (market data through API or CSV exports)
   - Yahoo Finance (crypto price history) (Recommended)
   - Kaggle (cryptocurrency datasets)

   Ensure the data includes key attributes like date, open price, close price, high, low, volume, and market cap. Clean and preprocess the data, ensuring it is time-indexed (daily or hourly). Document the data source, date range, and preprocessing steps.

2. **Exploratory Data Analysis (EDA):** Perform EDA to identify trends, seasonality, volatility, and outliers. Visualizations should include line plots of price over time, rolling averages, and volume trends. Calculate statistical summaries for key metrics. Provide insights into market behavior based on your analysis.

3. **Stationarity Testing:** Test the stationarity of the time series using the ADF test. Conduct tests on both raw and differenced data to check for trends and volatility clustering. Analyze the ADF statistics, p-values, and graphical representations, and conclude whether differencing is required.

4. **ARIMA Model Development:** Develop ARIMA models for price forecasting by determining the optimal (p, d, q) values using ACF and PACF plots. Train multiple ARIMA models and fine-tune hyperparameters. Ensure the modeling process is clearly documented, including the reasoning behind selected parameters and visualizations of ACF/PACF plots.

5. **Model Evaluation:** Evaluate model performance using error metrics such as Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). Compare predicted prices to actual data, visualize residuals, and interpret the model's accuracy and reliability.

6. **Forecasting & Visualization:** Generate future price projections for Ethereum for the next 30 days using the ARIMA model. Visualize forecasted prices with confidence intervals and highlight key trends. Provide clear explanations of the forecasted patterns.

Additional resources on ARIMA and Time Series Forecasting:

- ARIMA for Time Series Forecasting: A Complete Guide
- Understanding Time Series Forecasting with ARIMA
- Time Series Analysis using Python — ARIMA & SARIMAX Model Implementation — Stationarity Handling
- Using ARIMA to Predict Bitcoin Prices in Python in 2023
- Bitcoin price prediction! Using ARIMA and STDEV
- ARIMA Models for Stock Price Prediction: How to Choose the p, d, q Terms (Part 1/2)
- What is Time Series Analysis?
- Stock Market Forecasting with ARIMA, SARIMA, SARIMAX — A Complete Project A-Z
- Complete Exploratory Data Analysis and Feature Engineering in 3 Hours — Krish Naik

6.2 Task 02

Parameter-Efficient Supervised Fine-Tuning of LLaMA 3.2 (3B) on a Medical Chain-of-Thought Dataset

Objective: The objective of this task is to fine-tune the **LLaMA 3.2 (3B)** model using a **medical Chain-of-Thought (CoT) dataset** from Hugging Face while applying **parameter-efficient fine-tuning (PEFT)** techniques in **Unsloth** within **Kaggle Notebooks**. The goal is to enable the model to generate **step-by-step medical reasoning** and improve structured response generation. The fine-tuning process should be **tracked using Weights & Biases (wandb)**, and the fine-tuned **LoRA adapter and tokenizer** must be uploaded to **Hugging Face**. Model performance should be evaluated using the **ROUGE-L score** before and after fine-tuning.

Resources on LLaMA 3.2 Fine-Tuning and Medical Chain-of-Thought Datasets:

- LLaMA 3.2 (3B) Instruct Model – Unsloth
- Medical Chain-of-Thought Dataset – FreedomIntelligence
- Fine-Tuning Guide – Unsloth Documentation
- Unsloth Fine-Tuning Notebooks – GitHub Repository
- Llama3.2_(1B_and_3B)-Conversational.ipynb
- Fine Tune DeepSeek R1 — Build a Medical Chatbot

Task Requirements:

1. **Environment Setup:** Set up a fine-tuning environment using **Kaggle Notebooks**. Install the necessary libraries:

    - **Unsloth** (pip install unsloth automatically installs various libraries like transformers, pytorch, accelerate, bits and bytes, etc.)

    Verify GPU availability in Kaggle and configure a **Weights & Biases API key** for logging.

2. **Dataset Preparation:** Retrieve the **medical Chain-of-Thought dataset** from Hugging Face and format its structure.

    - **"Think" tags** → Represent step-by-step reasoning (Chain of Thought) under `<think>` ... `</think>`
    - **"Response" tags** → Represent the final answer to the medical query under `<response>` ... `</response>`

    Split the dataset into:

    - **100 rows** for validation.
    - The **remaining data** for training.

Perform preprocessing, ensuring proper text formatting and removal of unnecessary metadata.

3. **Model Selection & Fine-Tuning Strategy:** Load the **LLaMA 3.2 (3B)** model in its **quantized 4-bit format** using **Unsloth**. Apply **Low-Rank Adaptation (LoRA)** to fine-tune only selected model parameters. The fine-tuning strategy should include:
   - Using **supervised fine-tuning** on the Chain-of-Thought dataset.
   - Employing **gradient accumulation** for optimized GPU memory usage.
   - Selecting an appropriate **learning rate** and **batch size** for training stability.
4. **Fine-Tuning Process & Weights & Biases (wandb) Tracking:** Use **wandb** to log:
   - **Training loss trends**.
   - **Validation performance over epochs**.
   - **GPU memory consumption**.

   Monitor model improvement through **ROUGE-L scores** before and after fine-tuning.
5. **Model Saving & Deployment:** Save the fine-tuned components separately and upload them to Hugging Face:
   - **LoRA adapter** (fine-tuned weights).
   - **Tokenizer** (if modified).

   Provide clear instructions for:
   - Loading the **quantized base model**.
   - Attaching the **fine-tuned LoRA adapter**.
   - Generating medical responses using the fine-tuned model.

Expected Deliverables:

- A **Kaggle Notebook** containing:
  - The complete fine-tuning workflow.
  - Instructions for inference with the **fine-tuned LoRA adapter**.
- **Hugging Face Model Link** (where the LoRA adapter and tokenizer are uploaded).
- **wandb Logs** demonstrating training progress.
- **Comparison of ROUGE-L Scores** before and after fine-tuning.

7. Category B: Intermediate

7.1 Task 01

Python Basics (If Needed)
If you do not have any experience with Python, watch the following tutorial to get familiar with the basics. However, if you already know Python, you may skip this and proceed to the next task.

- Python Tutorial for Beginners - Learn Python in 1.5 Hours by Apna College

Understanding Google Colab and VS Code (If Needed)
If you are already familiar with Google Colab and VS Code, you may skip this task. Otherwise, watch the following videos to understand how to set up your development environment for Machine Learning projects.
Access Google Colab from here:

- Google Colab

Make a new notebook and watch this tutorial:

- Google Colab Tutorial for Beginners — Get Started with Google Colab by Doga Ozgon

To run code offline, install VS Code, See this video here.

- Getting started with Python in VS Code (Official Video) by Visual Studio Code

Project - Brain Tumor Segmentation with YOLO 11 and SAM2

This project involves using advanced computer vision techniques to perform brain tumor segmentation using YOLO 11 and SAM2 models.

Overview:

- YOLO 11 - A state-of-the-art real-time object detection model that identifies multiple objects in an image with high speed and accuracy.
- SAM2 (Segment Anything Model 2) - A segmentation model designed for precise object separation in an image.
- Segmentation - The process of partitioning an image into meaningful regions to detect and analyze objects.

Task Requirements:

- Understand how YOLO 11 and SAM2 work for segmentation tasks.
- Set up the necessary Python environment for implementation.
- Train and test the model using a dataset of brain tumor images.
- Evaluate the segmentation results and fine-tune the model if necessary.
- Document the implementation process, findings, and observations.

Follow the video below to complete this project:

- Yolo11 and SAM2 for custom Instance Segmentation by Code With Arohi Hindi

Machine Learning Overview
Once you have completed the Brain Tumor Segmentation Project, watch the following video for a general overview of Machine Learning. This will help you understand different ML techniques and fit into the broader AI landscape.

- Machine Learning for Everybody – Full Course by freecodecamp.org

This video provides a comprehensive introduction to Machine Learning concepts and implementations. It starts with an overview of data handling and Google Colab, followed by fundamental ML concepts such as features, classification, and regression. The video then explores different ML algorithms, including:

- **K-Nearest Neighbors (KNN)** – Concept and implementation.
- **Naive Bayes** – Theory and application in classification tasks.
- **Logistic Regression** – Understanding and implementation.
- **Support Vector Machines (SVM)** – Introduction and practical use.
- **Neural Networks** – Basics, TensorFlow usage, and model training.
- **Linear Regression** – Concept, implementation, and its neural network equivalent.
- **Unsupervised Learning** – K-Means Clustering and Principal Component Analysis (PCA) with implementations.

This video also includes hands-on demonstrations using TensorFlow and Python, making it an excellent resource for understanding both theory and practical implementation.

7.2 Task 02

Chapter 1 and 2 of Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow

This is the second task. You are required to read and understand the first two chapters of **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow** by Aurélien Géron.

You might already be familiar with some of these concepts, but revisiting them will reinforce your understanding and ensure you have a strong foundation for advanced topics. This book is highly beginner-friendly, with clear explanations and practical examples. Try reading it yourself first before looking for external resources.

Key Topics Covered

Chapter 1: The Machine Learning Landscape

- Basic Terminologies related to Machine Learning
- Supervised/Unsupervised/Batch/Online Learning
- Challenges related to Data and Algorithms
- Hyperparameter tuning and Model Selection

Chapter 2: End-to-End Machine Learning Project

- The Machine Learning pipeline
- Data exploration and visualization techniques
- Custom Transformers and Feature Scaling
- Fine-tuning hyperparameters using GridSearchCV and RandomizedSearchCV

Task Requirements:

- Read and understand Chapter 1 and 2, focusing on the key topics.
- Implement the code presented in the book and analyze the outputs.
- Take notes on important concepts (MS Word or handwritten).
- **Complete the Chapter 1 exercises and attach your Chapter 2 code with explanations** in report along with any notes you have made.
- For submission details, please refer to the **Submission Guidelines** section of this manual.

For additional explanations, you can visit these YouTube playlists:

- Hands-On Machine Learning by Shashank Kalanithi.
- Hands-On Machine Learning by San Diego Machine Learning.

For submission details, please refer to the **Submission Guidelines** section below in this manual.

8. Category C: Beginner

8.1 Task 01

This section is for those who have no prior experience with Machine Learning or Python. If you have learned any programming language before, such as C++, Python will be easier to pick up. However, if you are completely new to coding and don't know how to write a single line of code, don't worry—we've got you covered.

To start learning Python, we recommend watching the following videos and practicing the code along with them. Make notes, revise, and ensure you understand the basics. If you face issues downloading a compiler or software, you can use this online Python compiler:

- Online Python Compiler - Programiz

A compiler is a tool that understands and allows you to write and run code.

Python Learning Resources:

- Python Tutorial for Beginners — Learn Python in 1.5 Hours by Apna College
- Learn Python in Hindi in One Video by CodeWithHarry

Now that you have an understanding of loops, variables, functions, operators, etc., you can start practicing Python coding.

Additionally, visit this website on a PC:

- W3Schools Python Tutorial

On the left side of this website, you can see all the topics, including syntax details, functions, loops, operators, variables, dictionaries, and more. You can use this site as a reference whenever you want to learn more about Python.

Introduction to Google Colab

Now that you want to get into ML, the first thing to understand is **Google Colab**. It is an online compiler specifically designed for Machine Learning and Deep Learning tasks. Access Google Colab from here:

- Google Colab

Make a new notebook and watch this tutorial:

- Google Colab Tutorial for Beginners — Get Started with Google Colab by Doga Ozgon

Now that you have learned where to write Python code, you can focus on practicing Python regularly before moving on to Machine Learning.

8.2 Task 02

To mark the end of Task 1, there is a small reading task that is necessary. You are required to read Chapter 1 of Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron. This book was also used in my university days, and I want you to develop a strong understanding of its first chapter.

Chapter 1: The Machine Learning Landscape

- Basic Terminologies related to Machine Learning
- Supervised/Unsupervised/Batch/Online Learning
- Challenges related to Data and Algorithms
- Hyperparameter tuning and Model Selection

Task Requirements:

- Read Chapter 1 thoroughly.
- Take handwritten or digital notes on key concepts.
- If you don't understand any term, feel free to Google it or ask ChatGPT or any other AI tool for explanations with analogies.
- Solve all **19 exercise questions from Chapter 1**.
- Write to-the-point answers in MS Word.

First Machine Learning Project: Crop Prediction with Web Application

After reading Chapter 1 and completing the exercise questions, it's time to build your **first Machine Learning model** and deploy it as a simple web application!
This project is a crop prediction task where you will build an ML model that predicts the best crop to grow based on soil and weather conditions. This will introduce you to basic ML workflows, training models, and deploying applications.
To guide you through the process, follow this short YouTube playlist. The entire playlist is only **1 hour and 12 minutes long**, with simple explanations that are easy to follow:

- Machine Learning Project - Flask Web Application - Complete Python Project by AlgoChat

Make sure to follow along with the coding, understand each step, and note down key concepts. This will be your first practical Machine Learning experience!

If you have already completed this, then that is enough. However, now that you have developed a general understanding of ML and Python, you are encouraged to check out the tasks in other categories as well.

For submission details, please refer to the **Submission Guidelines** section at the end of this document.

9. Supplementary Tasks (Optional)

9.1 Installing DeepSeek on Your PC

As an optional task, we invite you to explore **DeepSeek** offline on your PC. This is a powerful language model that runs directly on your machine, offering coding assistance, text summarization, writing support, and more — all without needing an **internet connection**.

To get started, follow the step-by-step guide in this YouTube tutorial:

- Run your own AI (but in private) with NetworkChuck

In the video, Meta's LLaMA 3.2 model is installed, but you are free to explore other models available on the Ollama Model Library. You can install and experiment with any model that suits your needs, including **DeepSeek**.

This task is entirely voluntary and won't affect your main assignments, but it's a great way to expand your AI toolkit. Feel free to share your experience with us!

9.2 Creating Virtual Machines and Installing Ubuntu Linux

As another optional task, we encourage you to explore the world of **Linux** and **virtualization**. Installing **Ubuntu Linux** on a virtual machine (VM) allows you to experiment with a free, open-source operating system in a safe, isolated environment. Linux offers numerous advantages, including enhanced customization, robust security, a vast software repository, and a streamlined user experience optimized for developers and power users.

A **hypervisor** (like VirtualBox) is software that creates and manages virtual machines, enabling you to run multiple operating systems simultaneously on a single physical machine. Virtual machines provide the following benefits:

- **Isolation**: Test software, configurations, or unfamiliar OSes without risking your main system.
- **Flexibility**: Run Windows, Linux, or other OSes side-by-side.
- **Security**: Safely browse suspicious files or websites in a sandboxed environment.
- **Resource Efficiency**: Allocate specific CPU, memory, and storage resources to each VM.

To get started, follow these resources:

- Virtual Machines Tutorial by NetworkChuck (YouTube guide covering Kali Linux, Ubuntu, & Windows)

Required Downloads:

- Ubuntu 24.04 LTS: ubuntu.com/download/desktop
- VirtualBox: virtualbox.org
- VirtualBox Extension Pack: Download here

This task is optional and separate from your core assignments, but it's an excellent way to gain hands-on experience with virtualization and Linux. Share your progress or ask questions anytime!

10. Submission Guidelines

All project and theory task submissions must follow the format below to ensure clarity and consistency. Please review the steps and requirements carefully.

10.1 What to Include in Your Report?

Your submission should be compiled into a single PDF file containing:

1. **First Page** — The first page of the report must clearly show your Full Name, Phone Number and InternID.
2. **Project Overview** — A brief introduction explaining the task, its objectives, and the approach you used.
3. **Diagrams, Graphs, and Visuals** — Include any charts, plots, graphs or images that show how your data looks, model performance, or any other visualizations.
4. **Code with Explanations** — Add your code along with clear, short explanations for each step.
5. **Theory Tasks** — Attach your digital notes or images of handwritten notes relevant to theory-based tasks.
6. **Notes and References** — Include any additional notes you've made from tutorials, articles, or other study materials. Mention sources like YouTube videos or blogs.
7. **Video Demonstration** — Record a short video explaining your task and code. This can be a screen recording of your program running. Upload the video to YouTube and share the link in your report.
8. **GitHub Repository** — Upload your project files (code, data, etc.) to a GitHub repository and include the link in your report.

10.2 Additional Content

You are encouraged to attach any supplementary material related to your tasks — such as additional graphs, screenshots, or background research. However, please ensure that everything is compiled into a **single PDF file**.

10.3 File Naming Format

Name your submission file using this format:

```
[Category A B or C as CA CB or CC]-[Manual 1 as M1]-[INTERN ID].pdf
```

Example: `CA-M1-ARCH-2504-0000.pdf`

where CA stands for Category A, M1 stands for Manual 1 and on the last is Intern ID.

10.4 Submission Deadline

If you finish your tasks ahead of schedule, you may submit your report or zip file early. Moreover, if you'd like to delve deeper into ML, you are welcome to check the tasks of other categories also. Note that submissions must be sent via email to **submissions.archtech@gmail.com** by **the 27th of this month**. Kindly ensure that your report is thorough, organized, and professional. Good luck!

11. Evaluation Criteria

Submissions will be assessed based on the following criteria:

1. **Report Structure**
   - Logical organization with clear headings and sections.
   - Concise, well-articulated explanations.
   - Adherence to submission guidelines.
2. **Code Accuracy and Implementation**
   - Code must execute without errors and yield correct results.
   - Efficient use of libraries and frameworks.
   - Proper handling of exceptions and edge cases.
3. **Technical Explanation and Presentation**
   - Justification of model choices and methodology.
   - Use of visuals (charts, diagrams, plots) to enhance understanding.
   - Concise documentation of key processes.
4. **Code Readability and Documentation**
   - Clean, well-structured, and readable code.
   - Meaningful comments and function descriptions.
   - Proper use of version control (GitHub) with structured commits.
5. **Supplementary Resources**
   - Provide links to datasets used in your project.
   - Share your GitHub repository with properly documented code.
   - Include YouTube links to tutorials or references you used.
   - Record a video explaining your code or a screen recording of your implementation and upload it to YouTube. Share the link in your submission.

12. Acknowledgements

We express our heartfelt gratitude to all members of the ARCH Technologies team for their dedication and hard work. Your collective efforts drive our mission of building innovative AI solutions and fostering a culture of continuous learning.

We would also like to recognize our internship students for their enthusiasm and commitment. Your curiosity, determination, and willingness to learn are invaluable as we work together to make a better tomorrow. We look forward to seeing you excel and contribute to the future of AI.

Thank you all for being an essential part of this journey.