GENERIC ATTENTION FOR MULTI-TASK CONTROL TRANSFORMERS

A summer internship report submitted in partial fulfillment of the requirements forthe award of the degree of

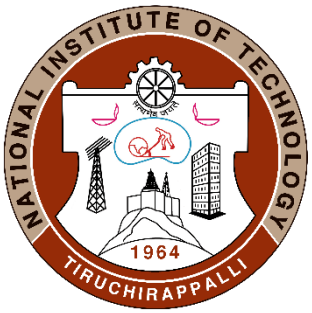
B.Tech.

in

Electrical and Electronics Engineering

By

ATHARVA RATHI (107121016)



DEPARTMENT OF
ELECTRICAL AND ELECTRONICS ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
TIRUCHIRAPPALLI-620015

AUG 2024

INTERNSHIP CERTIFICATE

TECHNISCHE UNIVERSITÄT DARMSTADT

Technische Universität Darmstadt | Karolinenplatz 5 | 64289 Darmstadt

Internship Completion Letter for Atharva Rathi

To Whom It May Concern:

We hereby confirm that Mr. Atharva Rathi, a student of NIT Trichy, completed a research internship at the PEARL Lab, Computer Science Department, at the Technische Universität Darmstadt (TU Darmstadt) from May 27, 2024, to August 20, 2024. The internship was conducted under the DAAD-WISE scholarship. He worked on Generic Attention for Multitask Control Transformers.

Sincerely,

Prof. Georgia Chalvatzaki

Interactive Robot Perception and Learning Lab (PEARL)

Prof. Georgia Chalvatzaki

Hochschulstra"se 10
64289 Darmstadt

Phone: +49 176 60 - 916821
Fax: +49 6151 16 - 7374
georgia.chalvatzaki@tu-darmstadt.de

Date
August 27, 2024

ACKNOWLEDGEMENTS

I would like to acknowledge and express our deepest gratitude to the following people for guiding us through this project, without whom this project would not have been completed.

I want to extend my heartfelt gratitude to Professor Georgia Chalvatzaki, a full-time professor at TU Darmstadt, for their invaluable guidance and support throughout my research internship. Your expertise and insightful feedback have been instrumental in shaping my understanding and approach to the research project.

I am also deeply thankful to Rickmer Kron, whose mentorship and encouragement made this experience truly enriching. Your patience and willingness to share your knowledge and skills have greatly contributed to my personal and professional growth. Both of your contributions have been essential to the successful completion of this internship, and I am sincerely grateful for the opportunity to learn from you. Thank you for your unwavering support and for inspiring me to strive for excellence.

Lastly, I am immensely thankful to my parents, professors and friends for their constant support and encouragement.

ABSTRACT

The project, titled “Generic Attention for Multitask Control Transformers,” focused on the complex task of visualizing attention mechanisms within these models. During the summer internship, significant advancements were made in the application of Generic Attention mechanisms to Multitask Control Transformers. Working in collaboration with a PhD student, we developed a comprehensive Visualization Pipeline that included a Graphical User Interface (GUI) and the capability to save essential plots. This tool enhanced users' ability to understand attention maps concerning model inputs intuitively.

The research centered around the BAKU multitask Transformer model, particularly examining 30 kitchen tasks from the Libero-Dataset. Multiple variations of attention maps were compared, and valuable data, such as attention weights and gradients, were efficiently extracted from the PyTorch model during inference.

A key element of the internship involved adapting the theoretical framework from the paper "Generic Attention-model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers" by Hila Chefer et al. from Vision Transformers to Control Transformers. This required a deep dive into the mechanics of Transformers and related explainability literature. The intern successfully highlighted the limitations of the generic attention method within the context of robotic control tasks. Additionally, a thorough analysis of various attention gradients was conducted to establish an appropriate weighting scheme, further enhancing the model's interpretability.

KEYWORDS: Generic Attention, Transformers, GUI, BAKU, Libero, Attention Maps, PyTorch

TABLE OF CONTENTS

Title **Page No.**

Contents

INTERNSHIP CERTIFICATE **2**

ACKNOWLEDGEMENTS **3**

ABSTRACT **4**

TABLE OF CONTENTS **5**

LIST OF FIGURES AND TABLES **7**

CHAPTER 1 INTRODUCTION **8**

1.1 About the Program 8

1.2 My Role 8

CHAPTER 2 FRAMEWORKS USED **9**

2.1 LIBERO Datasets Overview 9

2.2 EGL Library 10

2.3 Gymnasium Environment 10

2.4 MuJoCo Library 10

2.5 Robosuite 11

2.6 Tkinter Overview 11

2.6.1 Key Features of Tkinter 12

2.6.2 Using Tkinter for Robotics and Imitation Learning Applications 12

CHAPTER 3 PROJECT OVERVIEW **14**

3.1 Overview of Explainable AI and Its Importance in Robotics 14

3.1.1 BAKU: An Efficient Transformer for Multi-Task Policy Learning 14

3.1.2 Generic Attention-model Explainability 15

3.1.3 Challenges with Explainability Techniques in Robotics 16

3.2 Objective 17
3.3 Existing Methodologies 17
CHAPTER 4 METHODOLOGY 19
4.1 Explainability 19
4.1.1 Generic Attention and Relevancy Maps[1] 19
4.1.2 BAKU: An Efficient Transformer for Multi-Task Policy Learning[2] 21
4.1.3 LIBERO Dataset 24
4.1.4 Background 25
4.1.4.1 Imitation Learning 25
4.1.4.2 Multi-Task Behaviour Cloning 26
4.1.5 Graphical User Interface (GUI) for Visualizing Data 27
CHAPTER 5 SUMMARY AND CONCLUSION 30
5.1 Summary 30
5.2 Conclusion 30
CHAPTER 6 RESEARCH EXPERIENCE 31
6.1 Technical Skills Acquired from Working on the Project 31
6.2 Non-Technical Skills Acquired 33
REFERENCES 34

LIST OF FIGURES AND TABLES

List of Figures:

FIG NO.	FIGURE NAME	PAGE NO.
4.1	BAKU Transformer	23
4.2	GUI Front-End	29

CHAPTER 1

INTRODUCTION

1.1 About the Program

The DAAD WISE (Working Internships in Science and Engineering) Scholarship is a prestigious program offered by the German Academic Exchange Service (DAAD) aimed at providing Indian undergraduate students in the fields of science and engineering with the opportunity to gain research experience in Germany. This scholarship supports a 2–3-month internship at a German university or research institute, offering a chance to work on innovative projects under the guidance of expert mentors. The DAAD WISE Scholarship covers travel expenses, a monthly stipend, and health insurance, fostering academic and cultural exchange while enhancing the students' research skills and global exposure.

1.2 My Role

The PEARL Lab (Perception and Active Learning) at TU Darmstadt is a leading research group focused on advancing the fields of machine learning, robotics, and computer vision. The lab specializes in developing algorithms that enable machines to perceive, learn, and intelligently interact with complex environments. Under the leadership of Professor Georgia Chalvatzaki, the PEARL Lab is renowned for its contributions to reinforcement learning, imitation learning, and autonomous systems. The lab's interdisciplinary research bridges the gap between theoretical advancements and practical applications, aiming to create intelligent systems that can autonomously adapt and improve over time.

I worked with one of the PhD students in the PEARL Lab under the mentorship of Dr Georgia Chalvatzaki, where my focus was on extracting valuable information, like attention weights and gradients, from the Torch model during inference and efficiently storing them and developing a GUI (Graphical User Interface) including Front-End and Back-End for better explainability of multi-task control transformers used in robotics.

CHAPTER 2

FRAMEWORKS

USED

2.1 LIBERO Datasets Overview

LIBERO (Learning Imitation Behaviours through End-to-end Robust Optimization) datasets are designed to advance research in imitation learning, particularly in robotics. These datasets contain a comprehensive collection of multi-task learning data, including various simulated environments, agent behaviours, proprioceptive features, action sequences, attention maps, and camera views. The primary focus of LIBERO datasets is to provide the necessary data for training and evaluating machine learning models that aim to replicate human or expert behaviour in robotic tasks.

Components of LIBERO Datasets

LIBERO datasets typically include:

- **Proprioceptive Features**: Sensor data from the robot, including joint positions, velocities, forces, and other internal states.
- **Action Sequences**: The series of actions taken by the robot or agent during a task.
- **Attention Maps**: Data highlighting the importance of various input features to the robot's decision-making process.
- **Camera Views**: Visual data from different perspectives, including third person and egocentric (agent's) views.
- **Task Descriptions**: Metadata specifying the environment, task objectives, and agent characteristics.

These datasets are crucial for developing models that can generalize across different tasks, environments, and robotic agents.

Using LIBERO Datasets for Simulations

LIBERO datasets are highly versatile and can be used to run simulations in various environments and libraries, enabling researchers to test and refine their imitation learning models. Below is a guide on how to use these datasets with popular simulation libraries:

2.2 EGL Library

The EGL (Enhanced Graphics Library) is a powerful tool for rendering and simulating robotic environments. When combined with LIBERO datasets:

- **Data Integration**: The proprioceptive features, action sequences, and camera views from LIBERO can be directly loaded into an EGL-powered simulation environment.
- **Simulation**: Researchers can simulate robotic tasks by feeding the LIBERO data into the EGL framework, allowing for high-fidelity visualizations and real-time interaction with the simulated environment.
- **Imitation Learning**: The datasets can be used to train models that mimic expert behaviour within EGL, leveraging the graphical enhancements for better visual perception and action execution.

2.3 Gymnasium Environment

Gymnasium, an open-source toolkit for developing and comparing reinforcement learning algorithms, is compatible with LIBERO datasets in several ways:

- **Custom Environments**: Users can create custom Gymnasium environments using the task data from LIBERO datasets. These environments can simulate the same scenarios found in the datasets, providing a platform for testing and refining imitation learning models.
- **Agent Training**: The action sequences and proprioceptive features in LIBERO can be used as input data for training agents within the Gymnasium, allowing them to learn from expert demonstrations.
- **Benchmarking**: Researchers can use LIBERO datasets to benchmark their models' performance against standard Gymnasium tasks, ensuring robustness and generalization across different environments.

2.4 MuJoCo Library

MuJoCo (Multi-Joint dynamics with Contact) is a popular physics engine for simulating robotic environments, known for its accuracy and efficiency:

- **Physics-Based Simulations**: The proprioceptive features and action sequences from LIBERO can be used to create detailed physics-based simulations in MuJoCo, enabling realistic task execution and evaluation.
- **Task Replication**: Researchers can replicate the tasks described in LIBERO datasets

within MuJoCo, providing a controlled environment for studying the impact of different variables on task performance.

- **Imitation Learning**: MuJoCo's robust physics engine allows for the precise reproduction of expert demonstrations, making it an ideal platform for training imitation learning models using LIBERO data.

2.5 Robosuite

Robosuite is a flexible simulation framework designed specifically for robotics research, integrating seamlessly with LIBERO datasets:

- **Environment Setup**: Users can set up robotic tasks in Robosuite using the environment and task descriptions from LIBERO datasets. This setup allows for the direct application of data to simulate complex robotic behaviours.
- **Multi-Task Learning**: Robosuite's support for multi-task scenarios makes it particularly well-suited for using LIBERO datasets, which contain data for various tasks. Researchers can explore how models trained on one task generalize to others.
- **Imitation Learning**: The rich visual and proprioceptive data in LIBERO datasets can be utilized to train robots within Robosuite to imitate expert behaviours, testing their ability to perform tasks in a simulated environment before deploying them in real-world applications.

The LIBERO datasets are a valuable resource for advancing imitation learning in robotics, providing extensive data for training, testing, and validating models across different tasks and environments. By integrating these datasets with powerful simulation libraries such as the EGL Library, Gymnasium Environment, MuJoCo Library, and Robosuite, researchers can create sophisticated simulations that replicate real-world scenarios, ultimately leading to the development of more capable and adaptable robotic systems.

2.6 Tkinter Overview

Tkinter is the standard GUI (Graphical User Interface) toolkit for Python, widely used for developing desktop applications. Its simplicity, flexibility, and integration with Python make it an ideal choice for creating interactive and user-friendly interfaces. Tkinter allows developers to design complex interfaces with minimal code, offering a range of widgets like buttons, labels, sliders, and canvases that can be customized to meet specific application needs.

2.6.1 Key Features of Tkinter

- **Widgets**: Tkinter provides a wide variety of widgets (e.g., buttons, labels, text boxes, sliders) that can be used to build a fully functional GUI.
- **Layouts**: Tkinter supports multiple layout management options, including grid, pack, and place, making it easy to organize and arrange widgets within the application window.
- **Event Handling**: Tkinter allows for efficient event handling, enabling the creation of responsive applications that react to user input.
- **Cross-Platform**: Tkinter is cross-platform, meaning GUIs created with it will work on Windows, macOS, and Linux without requiring code changes.

2.6.2 Using Tkinter for Robotics and Imitation Learning Applications

Tkinter is particularly useful in robotics and imitation learning for developing GUIs that facilitate the visualization, control, and analysis of simulation data. Below are several ways Tkinter can be applied in this domain:

1. Data Visualization

Tkinter can be used to create GUIs that display various types of data generated during robotic simulations, including:

- **Attention Maps**: GUI elements such as canvases and labels can be used to display attention maps, showing how different input features influence the robot's decision-making process.
- **Proprioceptive and Action Features**: Graphs and plots can be embedded within a Tkinter window to visualize proprioceptive features and action sequences over time, helping researchers analyze the robot's internal state and behavior.
- **Camera Views**: Tkinter can display images from third-person and egocentric (agent's) camera views, providing a visual context for the robot's actions during a task.

2. Simulation Control and Navigation

Tkinter can also be used to design control panels for navigating through different timesteps of a simulation:

- **Sliders**: Tkinter's slider widgets can be used to scroll through time steps, updating all displayed visualizations in real time as the slider is moved.
- **Buttons**: Control buttons can be implemented for loading datasets, running simulations, saving visualizations, and switching between different views or modes within the application.

3. Data Management

Tkinter provides functionality for managing and interacting with datasets, including:

- **File Browsing**: Tkinter's file dialogue widgets allow users to browse and select directories or files containing simulation data, making it easy to load different datasets into the GUI.
- **Combo boxes and Lists**: Users can select from a list of available tasks or events stored in a dataset, streamlining the process of switching between different simulations or data files.

4. Interactive Visualization of Robotics Simulations

Tkinter enables the creation of interactive dashboards for real-time visualization of robotic simulations:

- **Multi-Window Support**: Tkinter allows for the creation of multiple windows within a single application, enabling users to open detailed plots or visualizations in separate windows while maintaining a clear overview of the main simulation.
- **Resizable Content**: Tkinter supports resizable widgets, ensuring that the GUI can adapt to different screen sizes and resolutions, which is crucial for maintaining the clarity of visualizations.
- **Real-Time Updates**: The GUI can be programmed to update visualizations and data displays in real-time as the simulation progresses, providing users with immediate feedback on the robot's performance.

5. Integration with Other Libraries

Tkinter can be easily integrated with other Python libraries to enhance its functionality:

- **Matplotlib and Seaborn**: These libraries can be used alongside Tkinter to create complex plots and graphs that are then embedded within the Tkinter GUI.
- **Pillow**: The Python Imaging Library (Pillow) can be used to handle and display images within the Tkinter interface, such as the camera views from robotic simulations.
- **Pickle**: Tkinter GUIs can be designed to load and process data stored in pickle files,

allowing for the seamless visualization of large datasets without the need for extensive preprocessing.

CHAPTER 3

PROJECT

OVERVIEW

3.1 Overview of Explainable AI and Its Importance in Robotics

Explainable AI (XAI) refers to methods and techniques that make the decision-making processes of AI models transparent and understandable to humans. As AI systems grow in complexity, particularly with deep learning models like Transformers, they often function as "black boxes," making decisions that are difficult to interpret. XAI aims to open this black box, providing insights into how models arrive at their predictions or actions, which is crucial for building trust, ensuring accountability, and improving model performance.

In the field of robotics, especially when using Transformer-based control models, explainability is of paramount importance. Robots often operate in dynamic and safety-critical environments where understanding the rationale behind their actions is essential. For example, in robotic control tasks, where a model must make real-time decisions, knowing why a particular action was chosen can help identify potential errors, improve safety, and refine the model for better performance.

Transformer-based models, widely used for their powerful ability to handle complex sequences of data, introduce additional challenges due to their intricate attention mechanisms. Explainability in these models allows researchers and engineers to visualize and interpret the attention maps, understand how different inputs influence decisions, and ensure that the model's focus aligns with human expectations. This is particularly critical in applications such as autonomous systems, where an unexplained decision could lead to significant consequences.

3.1.1 BAKU: An Efficient Transformer for Multi-Task Policy Learning

The research addresses the long-standing challenge of training generalist policies for physical agents in robotics, where data acquisition is significantly more challenging and

costly compared to fields like computer vision and natural language processing. Traditional approaches to multi-task policy training often involve collecting large amounts of data through teleoperators, but these policies tend to be inefficient, often underperforming compared to single-task policies.

The proposed solution, BAKU, is a new architecture specifically designed for efficient multi-task policy learning in robotics, even in data-scarce scenarios. BAKU incorporates three key features:

1. **Transformer Encoder**: Fuses information from multiple modalities, such as vision and language, while incorporating temporal context.

2. **FiLM-Conditioned Visual Encoder**: Adapts the visual encoder to specific tasks, enabling the model to learn task-specific representations.

3. **Action Prediction Head**: Separated from the observational encoding trunk, allowing BAKU to be easily integrated with state-of-the-art action generation models.

In extensive experiments, BAKU demonstrated significant performance improvements, particularly on the LIBERO dataset:

- **LIBERO Dataset Performance**: BAKU achieved a 90% average success rate, marking a 36% absolute improvement over previous state-of-the-art methods, setting a new benchmark in multi-task learning performance.

- **Overall Performance**: Across 129 simulated tasks, BAKU exhibited an 18% absolute performance improvement compared to prior algorithms.

The architecture's effectiveness was further validated through real-world robotic tasks, where BAKU outperformed existing methods, particularly due to its innovative use of action chunking and a multimodal action head, as revealed in the ablation analysis.

.

3.1.2 Generic Attention-model Explainability

Transformers are increasingly dominating multi-modal reasoning tasks, such as visual question answering, achieving state-of-the-art results thanks to their ability to contextualize information using self-attention and co-attention mechanisms. These attention modules also play a role in other computer vision tasks including object

detection and image segmentation. Unlike Transformers that only use self-attention, Transformers with co-attention are required to consider multiple attention maps in parallel to highlight the information that is relevant to the prediction in the model's input. In this work, we propose the first method to explain prediction by any Transformer-based architecture, including bi-modal Transformers and Transformers with co-attentions. We provide generic solutions and apply these to the three most used of these architectures: (i) pure self-attention, (ii) self-attention combined with co-attention, and (iii) encoder-decoder attention. We show that our method is superior to all existing methods which are adapted from single modality explainability.

3.1.3 Challenges with Explainability Techniques in Robotics

Multi-modal Transformers are transforming computer vision by enabling models to handle multiple tasks without the need for task-specific training. Traditional computer vision models are trained for specific tasks with fixed outputs. However, recent advancements by Radford et al. and Ramesh et al. have shown that models trained on both text and images using Transformers can perform various tasks at state-of-the-art accuracy without additional training.

Radford's model encodes text and images separately using Transformers and then applies a contrastive loss to align them. Ramesh's approach, on the other hand, concatenates image and text representations and processes them together with a Transformer to generate images from text descriptions.

These models integrate text and images in different ways, leading to challenges in explainability. Existing methods for explaining Transformer models typically rely on self-attention mechanisms, but multi-modal Transformers often use other forms of attention, making them difficult to interpret with current explainability techniques.

Developing generalist policies that allow robots to solve multiple tasks is a longstanding challenge in decision-making and robotics. While there have been significant advancements in computer vision and natural language processing, robotics lags. The main reason is the difficulty in obtaining large-scale data. Unlike vision and language data, which can be easily collected from the internet, robotics data requires physical interaction with the environment. This makes data collection for robotics more time-consuming and expensive, hindering progress in creating versatile robotic systems.

3.2 Objective

The objective of this project is to create a GUI to visualize various features, such as action, proprioceptive features, and attention maps across multiple timesteps in the Libero dataset in robotics, is to provide researchers and engineers with an intuitive and interactive tool to analyze and interpret complex datasets. By integrating these diverse data streams into a single, cohesive interface, the GUI enables users to observe correlations, patterns, and anomalies in the behaviour of robotic systems across different tasks. This visualization aids in debugging, performance evaluation, and the development of more efficient algorithms, ultimately contributing to advancements in the field of robotics.

3.3 Existing Methodologies

Layer-wise Relevance Propagation (LRP) is a popular explainability method used to understand the decisions made by neural networks, including multi-task control transformers like BAKU applied to the Libero dataset. LRP works by backpropagating the model's output relevance scores through the network layers, assigning relevance values to individual input features. This helps in identifying which parts of the input data are most responsible for the model's predictions.

When applied to multi-task control transformers, LRP can be particularly useful for analysing how the model distributes its attention across various tasks and data types, such as action sequences, proprioceptive features, and attention maps. By visualizing the relevance scores across these different inputs, researchers can gain insights into the internal decision-making process of the model. For example, LRP can help identify whether the model is correctly focusing on the most relevant inputs for a given task or if it's being influenced by irrelevant features.

However, LRP has several flaws in this context. One of the main issues is that LRP tends to produce noisy explanations, especially in deep and complex models like transformers. The relevance scores can sometimes highlight irrelevant or redundant features, leading to misleading interpretations of the model's behaviour. This noise can be particularly problematic when dealing with multi-task scenarios where the model needs to distinguish between various tasks with similar input features.

Another limitation of LRP is that it assumes a linear relationship between the input features and the model's output, which might not hold in complex, non-linear models like transformers. This assumption can result in oversimplified explanations that fail to capture the true nature of the model's decision-making process. Additionally, LRP does not account for interactions between different input features, which can be crucial in multi-task learning where the model's decisions are often influenced by complex, interdependent factors.

Finally, LRP's explanations are sensitive to the choice of the underlying neural network architecture and the specific task at hand. This means that the same input might be assigned different relevance scores depending on the model and task, making it difficult to generalize the findings across different scenarios.

In summary, while LRP offers valuable insights into the decision-making process of multi-task control transformers like BAKU on the Libero dataset, its inherent noise, assumptions, and sensitivity to model architecture limit its effectiveness as a comprehensive explainability tool in this context.

CHAPTER 4

METHODOLOGY

4.1 Explainability

4.1.1 Generic Attention and Relevancy Maps[1]

The project describes a method for generating relevancy maps in multi-modal attention networks, specifically focusing on interactions between two different types of input data, such as text and images. The goal is to explain how different parts of the input data contribute to the model's decisions by analysing the attention layers within the model. This method is versatile and can be applied to any Transformer-based architecture, not just those involving text and images.

Key Concepts:

1. **Input Modalities**: The method considers two types of inputs, which in this context are text and image tokens. These tokens represent the individual pieces of data that the model processes.

2. **Attention Interactions**: The model has four types of attention interactions:

    - **Self-Attention (Att and Aii)**: These interactions occur within a single modality (e.g., text with text, image with image).

    - **Multi-Modal Attention (Ati and Ait)**: These interactions occur between different modalities (e.g., how text tokens influence image tokens and vice versa).

3. **Relevancy Maps**: For each type of attention interaction, a relevancy map is constructed. These maps (Rtt, Rii for self-attention and Rti, Rit for multi-modal attention) track the influence of different tokens on each other throughout the model's layers.

Method Steps:

1. **Relevancy Initialization**:
    - For self-attention, the relevancy maps are initialized with identity matrices because each token initially only contains information about itself.
    - For multi-modal interactions, the relevancy maps are initialized to zeros since there is no initial context shared between different modalities.

$$\mathbf{R}^{ii} = \mathbb{I}^{i \times i}, \quad \mathbf{R}^{tt} = \mathbb{I}^{t \times t} \tag{1}$$
$$\mathbf{R}^{it} = \mathbf{0}^{i \times t}, \quad \mathbf{R}^{ti} = \mathbf{0}^{t \times i} \tag{2}$$

2. **Relevancy Update Rules**:
    - As the model processes the input data, the attention layers update the relevancy maps. The method leverages the attention maps generated during this process to adjust the relevancy scores.
    - The attention mechanism calculates attention scores using queries (Q), keys (K), and values (V) matrices. The attention map A connects the query tokens with the key tokens, defining the relationships between them.

$$\mathbf{A} = softmax(\frac{\mathbf{Q} \cdot \mathbf{K}^\top}{\sqrt{d_h}}) \tag{3}$$
$$\mathbf{O} = \mathbf{A} \cdot \mathbf{V} \tag{4}$$

3. **Aggregation of Relevancies**:
    - The method accumulates relevancy scores across multiple layers, accounting for the residual connections that maintain some of the original token information while adding new contextual information from the attention process.

$$\bar{\mathbf{A}} = \mathbb{E}_h((\nabla \mathbf{A} \odot \mathbf{A})^+) \tag{5}$$

$$\mathbf{R}^{ss} = \mathbf{R}^{ss} + \bar{\mathbf{A}} \cdot \mathbf{R}^{ss} \tag{6}$$
$$\mathbf{R}^{sq} = \mathbf{R}^{sq} + \bar{\mathbf{A}} \cdot \mathbf{R}^{sq} \tag{7}$$

    - For multi-modal attention, the relevancy maps are normalized to ensure that both self-attention (within the same modality) and cross-modal interactions (between different modalities) are properly balanced.

4. **Normalization and Final Updates**:

    - Self-attention maps are normalized to maintain a balance between the influence of each token on itself and the contextual information from other tokens.

$$\hat{\mathbf{S}}^{xx}_{m,n} = \sum_{k=1}^{|x|} \hat{\mathbf{R}}^{xx}_{m,k} \tag{8}$$

$$\bar{\mathbf{R}}^{xx} = \hat{\mathbf{R}}^{xx} / \hat{\mathbf{S}}^{xx} + \mathbb{I}^{x \times x}, \tag{9}$$

    - The final relevancy maps are updated by considering both self-attention and cross-modal attention, ensuring that the interactions between different modalities are accurately captured.

This method provides a detailed approach to understanding how multi-modal attention networks, like those used in Transformers, combine and process information from different input types, such as text and images. Generating relevancy maps makes it possible to visualise and explain the model's decision-making process, making the model's operations more transparent. This flexible method can be applied to any multi-modal setup, extending beyond just two modalities.

4.1.2 BAKU: An Efficient Transformer for Multi-Task Policy Learning[2]

The design of multi-task learning algorithms requires careful consideration of model architecture and component selection. This often leads to complex models where the role of individual components can be unclear. In this work, we introduce BAKU, a streamlined architecture for multi-task policy learning, and conduct a comprehensive ablation study across various existing multi-task learning architectures. BAKU is structured into three key components: Sensory Encoders, an Observation Trunk, and an Action Head.

1. Sensory Encoders

BAKU's sensory encoders process raw sensor inputs from various modalities into useful feature representations. These modalities include:

- **Vision**: A ResNet-18 visual encoder processes scene images, enhanced with a FiLM (Feature-wise Linear Modulation) layer to integrate task-specific information.
- **Proprioception**: A two-layer multilayer perceptron (MLP) encoder processes robot proprioception data.

- **Task Instructions**: Task instructions, provided as text or goal images, are processed using a 6-layer MiniLM text encoder.

To ensure compatibility across modalities, the outputs from all encoders are projected to the same dimensionality through additional MLP layers.

2. Observation Trunk

The Observation Trunk combines the encoded inputs from all sensory modalities. Two variants of the trunk network are explored:

- **Multilayer Perceptron (MLP)**: Encoded inputs are concatenated into a single feature vector and passed through an MLP. For historical observations, inputs from all time steps are concatenated.
- **Transformer**: Each encoded input is treated as an observation token and passed through a transformer decoder network. A learnable action token is appended to the observation tokens to predict actions. For historical observations, a separate action token is added for each time step, with a causal mask ensuring actions are predicted based on past observations.

Both variants output action feature vectors, which are then passed to the Action Head for action prediction.

3. Action Head

The Action Head is responsible for predicting actions based on the action feature vectors from the Observation Trunk. BAKU includes five variants of action heads:

- **Vanilla MLP**
- **Gaussian Mixture Model (GMM)**
- **Behavior Transformer (BeT)**
- **Vector-Quantized Behavior Transformer (VQ-BeT)**
- **Diffusion Policy**

To address the temporal correlation in robot movements and mitigate the covariate shift seen in low-data imitation learning, action chunking with exponential temporal smoothing is employed. Unlike previous approaches that decode actions for each time

step separately, BAKU predicts the action chunk as a single concatenated vector, simplifying the process and improving performance.

4. Final Architecture

The final BAKU architecture integrates a FiLM-conditioned ResNet-18 vision encoder, an MLP encoder for robot proprioception, and a pre-trained text encoder for task instructions. For environments with multiple camera views, a common visual encoder is used across all views. The Observation Trunk utilizes a causal transformer decoder architecture, and the base version of BAKU employs an MLP action head with action chunking and temporal smoothing for smoother motion generation.

Summary of Model Parameters:

- **Sensory Encoders**: ~2.1M parameters
- **Observation Trunk**: ~6.5M parameters
- **Action Head**: ~1.4M parameters
- **Total Model Size**: ~10M parameters

Through extensive experimentation, BAKU demonstrates strong performance across various multi-task learning scenarios, offering a simplified yet effective architecture for multi-task policy learning in robotics.

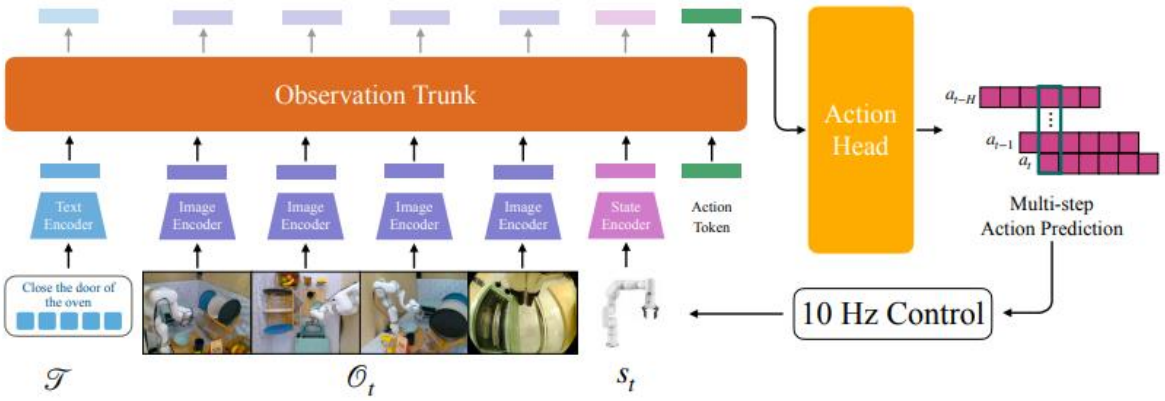


Fig 4.1: BAKU Transformer

4.1.3 LIBERO Dataset

The LIBERO-90 dataset is a comprehensive benchmark consisting of 90 diverse manipulation tasks designed for evaluating multi-task policy learning in robotics. The dataset offers a challenging environment for testing the capabilities of learning algorithms across a wide range of robotic manipulation scenarios.

4.1.3.1 Key Features of LIBERO-90:

- **Task Variety**: The dataset includes 90 distinct manipulation tasks, covering a broad spectrum of robotic activities. These tasks are specifically curated to test the generalization and learning capabilities of multi-task policies.
- **Demonstrations**: For each task in the LIBERO-90 dataset, 50 demonstrations are provided. These demonstrations serve as a valuable resource for imitation learning, allowing models to learn from expert behaviour. The demonstrations include visual and proprioceptive data, which are crucial for training effective policies.
- **Input Modalities**:
    - **Third Person View**: The dataset includes images captured from a third-person perspective, providing an external view of the robot and its environment.
    - **Gripper Camera View**: Alongside the third-person view, the dataset also provides images from a camera mounted on the robot's gripper. This egocentric perspective offers a detailed view of the objects being manipulated.
    - **Proprioception**: In addition to visual inputs, the dataset includes proprioceptive data, which captures the robot's internal states, such as joint angles, velocities, and forces. This information is critical for precise manipulation tasks.
- **Image Specifications**: The images in the LIBERO-90 dataset are of size 128×128

pixels, providing a balance between visual detail and computational efficiency.

- **Simulated Environment**: The tasks in LIBERO-90 are simulated, allowing for controlled experiments and consistent evaluations of policy performance across different tasks. The simulation environment replicates various real-world scenarios, enabling researchers to train and test robotic policies in a versatile setup.

4.1.3.2 Applications in Multi-Task Learning:

- **Policy Learning**: The LIBERO-90 dataset is used to train and evaluate multi-task policies, particularly in the context of imitation learning. The diversity of tasks in the dataset makes it an ideal benchmark for developing policies that can generalize across different manipulation tasks.
- **Benchmarking**: LIBERO-90 serves as a standard benchmark for comparing the performance of different multi-task learning algorithms. It provides a consistent framework for evaluating how well a model can perform across a wide range of robotic tasks.
- **Simulation Setup**: In experiments using the LIBERO-90 dataset, the simulation is set up with a combination of visual and proprioceptive inputs. Models are trained using the provided demonstrations, and their performance is assessed through policy rollouts in the simulated environment.

Overall, the LIBERO-90 dataset plays a critical role in advancing the field of multi-task policy learning by offering a robust and diverse set of tasks for developing and testing robotic policies.

4.1.4 Background

4.1.4.1 Imitation Learning

Imitation Learning is a technique aimed at developing a behaviour policy π^b by learning from an expert's behaviour, either directly from the expert policy π^b or from trajectories generated by the expert T^e . In the context of this project, we focus on a specific setting where the agent only has access to observation-based trajectories, denoted as,

$$\mathcal{T}^e \equiv \{(o_t, a_t)_{t=0}^T\}_{n=0}^N$$

where N is the number of trajectory rollouts, and T is the number of timesteps per episode.

This setting is chosen for its practical applicability in real-world scenarios, where obtaining both observations and actions from expert or near-expert demonstrators is feasible. This approach is consistent with recent advancements in the field, which emphasize learning from observation-based data due to its availability and practicality in real-world tasks.

4.1.4.2 Multi-Task Behaviour Cloning

Multi-Task Behaviour Cloning (BC) is an extension of the traditional Behaviour Cloning approach, designed to handle multiple tasks simultaneously. In standard Behaviour Cloning, the goal is to solve a maximum likelihood problem, where the objective is to train the policy π^b to mimic the expert's actions by minimizing the difference between the actions taken by the expert and those predicted by the policy. The loss function for Behaviour Cloning in a single-task setting is given by:

$$\mathcal{L}^{BC} = \mathbb{E}_{(o^e, a^e) \sim \mathcal{T}^e} \|a^e - \pi^{BC}(o^e)\|^2$$

In a multi-task scenario, the policy must be able to generalize across various tasks. To achieve this, the policy π^b is conditioned not only on the observation o^e but also on a goal variable g^e, which represents the specific task to be performed. This goal variable could be a textual description of the task or a goal image. The loss function for multi-task Behavior Cloning is thus modified as follows:

$$\mathcal{L}^{BC} = \mathbb{E}_{(o^e, a^e, g^e) \sim \mathcal{T}^e} \|a^e - \pi^{BC}(o^e|g^e)\|^2$$

This formulation allows the policy to learn to predict actions that are appropriate not only for the observed state but also for the specific task or goal. By conditioning on the goal, the policy can effectively learn to perform a wide range of tasks, making it suitable for applications in multi-task learning environments.

In summary, **Imitation Learning** in this project focuses on learning from observation-based trajectories, a practical and widely applicable approach in real-world settings. **Multi-task behaviour Cloning** extends this by incorporating task-specific goals into the learning process, enabling the policy to generalize across multiple tasks and making it a powerful tool for developing versatile and effective behaviour policies in complex, multi-task environments.

4.1.5 Graphical User Interface (GUI) for Visualizing Data

Here is a detailed overview of a graphical user interface (GUI) created using Tkinter, designed to facilitate the visualisation of robotics data stored in pickle files. This GUI is intended to analyse data from various simulation tasks, providing a user-friendly platform to explore the impact of input tokens on output actions across multiple time steps.

1. Directory Path Input

The GUI begins with a directory path input section that allows users to specify the location of the event data files. This section includes:

- **Directory Path Entry**: A text entry box where users can manually input the path to the directory containing pickle files.
- **Browse Button**: This button opens a file dialogue for selecting the directory, which automatically populates the path in the entry box.
- **Load Button**: After specifying the directory, users can click this button to load all pickle files from the directory into a drop-down menu.

2. Event Selection

Once the directory is loaded, users can select an event file to visualise:

- **Drop-Down Menu**: Displays the list of available pickle files in the selected directory.
- **Load Event Button**: Loads the selected event file and triggers the display of relevant visualisations.

3. Visualization Components

When an event file is selected, the GUI presents various visualisations and information:

- **Generic Attention Maps**: The GUI displays two attention maps that show the importance of input tokens on the action token and action features token across all timesteps.
- **Attention Graphs**: Four graphs depict the first four values from the action row in the generic attention map, providing insight into the influence of specific input tokens

over time.

- **Proprioceptive and Action Features Graphs**: These graphs show the changes in proprioceptive features and action features throughout the task, allowing users to track how these features change over timesteps of the simulation.

- **Camera Views**:
    - **Third Person View**: Displays an image representing the third-person camera view at each timestep.
    - **Agent View**: Displays an image representing the agent's viewpoint at each timestep.
- **Text Information Box**: Provides detailed information for each timestep, including:
    - **Environment Name**: The simulation environment.
    - **Task Name**: The specific task being performed.
    - **Agent Description**: Details about the agent (e.g., BAKU control transformer).
    - **Suite Used**: The suite or benchmark used (Libero in this case).
    - **Goal Status**: Indicates whether the goal has been achieved at the current timestep.

4. Time Step Navigation

- **Slider**: A slider allows users to navigate through different timesteps of the task. As the slider moves, the displayed visualizations, camera views, and text information update accordingly.

5. Save Functionality

- **Save as PNG Button**: Users can save the displayed plots, graphs, and text information by clicking this button.

- **Folder Selection**: Upon clicking, users are prompted to specify a folder name and path.
- **File Saving**: The GUI saves the visualizations as PNG images and the descriptive text as a text file in the specified folder.

6. Additional Features:

- **Opening Plot Windows:** Clicking the "Attn Map" or "Action Tokens" buttons opens a new window that displays an attention map or action tokens plot that is currently accepted by visualisation standards and doesn't consider all the layers of the transformer and treat all heads equivalent to each other.

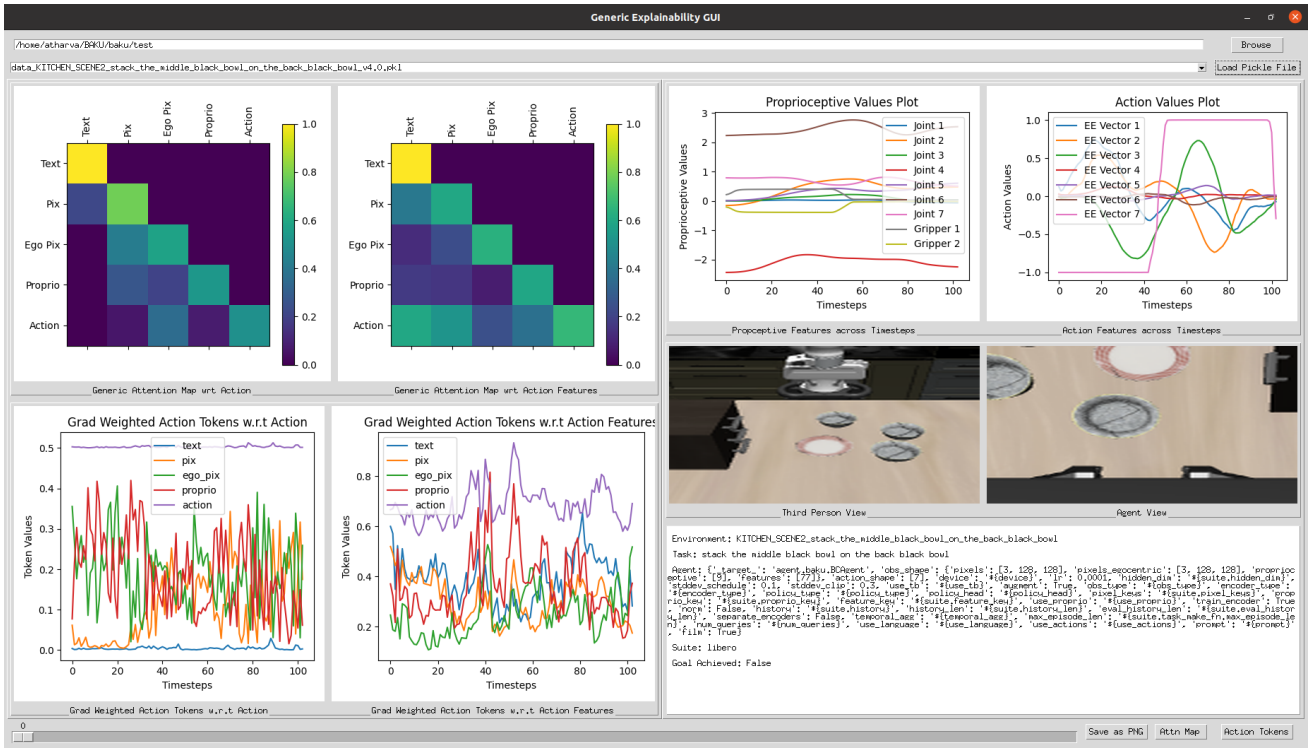


Fig 4.2: GUI Front-End

CHAPTER 5

SUMMARY AND CONCLUSION

5.1 Summary

In this work, we used BAKU, a transformer-based architecture designed to enhance multi-task policy learning across various simulated and real-world domains. Our results demonstrated that BAKU outperforms prior state-of-the-art methods in many tasks, showcasing its effectiveness in diverse environments. The LIBERO datasets, including LIBERO-90 and LIBERO-10, offer a comprehensive suite of benchmarks for evaluating multi-task policy learning in robotics. These datasets are essential for developing and testing models like BAKU, which are designed to handle a wide range of tasks, from simple manipulations to complex, long-horizon activities.

The Tkinter-based GUI offers a robust tool for visualizing and analysing event data from robotics simulation tasks. It integrates directory browsing, event selection, dynamic visualization, and data-saving capabilities, making it an efficient and user-friendly platform for exploring multi-task learning scenarios in robotics.

5.2 Conclusion

Our method provides a detailed approach to understanding how multi-modal attention networks, like those used in Transformers, combine and process information from different input types, such as text and images. Generating relevancy maps makes it possible to visualise and explain the model's decision-making process, making the model's operations more transparent. This flexible method can be applied to any multi-modal setup, extending beyond just two modalities.

In conclusion, the development of a GUI for visualizing various features within the Libero dataset offers a powerful tool for researchers and engineers in robotics. By integrating diverse data streams into an interactive interface, the GUI enhances the ability to analyze and interpret complex behaviours, supporting debugging, performance evaluation, and the creation of more efficient algorithms. This advancement is poised to significantly contribute to the ongoing development and refinement of robotic systems.

CHAPTER 6

RESEARCH EXPERIENCE

6.1 Technical Skills Acquired from Working on the Project

1. Data Processing and Management

- **Handling Complex Datasets:** Gain experience in managing and processing large-scale datasets like the LIBERO dataset, which includes various data types such as actions, proprioceptive features, and attention maps.
- **Preprocessing Data:** Learn techniques for data cleaning, normalization, and transformation to prepare data for training and visualization.

2. Graphical User Interface (GUI) Development

- **GUI Design and Implementation:** Develop skills in designing and implementing a user-friendly interface using tools like Tkinter or other Python-based GUI frameworks.
- **Data Visualization:** Learn to visualize complex data streams (e.g., actions, attention maps) cohesively and interactively, aiding in analysis and interpretation.

3. Machine Learning and Imitation Learning

- **Behavior Cloning:** Understand and implement behaviour cloning algorithms, particularly in multi-task settings, and learn to condition action predictions on various goal modalities.
- **Model Evaluation:** Gain experience in training machine learning models, evaluating their performance, and fine-tuning hyperparameters to optimize results.

4. Simulation and Robotics

- **Simulation Environments:** Work with simulation environments like MuJoCo, Gymnasium, and Robosuite to simulate robotic tasks and collect data for training.
- **Long-Horizon Task Learning:** Learn techniques for training models to perform long-horizon tasks, where sequences of actions need to be planned and executed over extended periods.

5. Programming and Software Development

- **Python Programming:** Enhance proficiency in Python, especially in areas like object-oriented programming, data manipulation, and integration of machine learning libraries.
- **Software Architecture:** Learn to design scalable and modular software architectures that can handle multi-task learning and real-time data visualization.

6. Robotics and Control Systems

- **Proprioceptive Feature Analysis:** Gain knowledge in interpreting and utilizing proprioceptive features, which are crucial for controlling robotic movements.

7. Research and Development

- **Experimentation and Ablation Studies:** Conduct ablation studies to analyze the impact of different architectural and design choices on the performance of multi-task policies.
- **Technical Writing and Documentation:** Develop skills in documenting experiments, writing technical reports, and communicating findings effectively

6.2 Non-Technical Skills Acquired

1. **Problem-Solving**
- Decompose complex problems and critically analyze data to make informed decisions.

2. **Collaboration**
- Enhance teamwork and communication skills across interdisciplinary teams.

3. **Project Management**
- Develop time management, milestone tracking, and task prioritization skills.

4. **Adaptability**
- Learn to handle uncertainty, quickly adapt to new tools, and adjust to evolving project needs.

5. **Creativity**
- Cultivate innovative thinking and prototyping skills for exploring new ideas.

6. **Resilience**
- Develop perseverance in overcoming challenges and continuously improving your work.

7. **Networking**
- Build professional relationships and explore career development opportunities.

REFERENCES

[1] H. Chefer, S. Gur, and L. Wolf, “Generic Attention-model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers,” Mar. 29, 2021, *arXiv*: arXiv:2103.15679. Accessed: Sep. 02, 2024. [Online]. Available: http://arxiv.org/abs/2103.15679

[2] S. Haldar, Z. Peng, and L. Pinto, “BAKU: An Efficient Transformer for Multi-Task Policy Learning,” Jul. 16, 2024, *arXiv*: arXiv:2406.07539. Accessed: Sep. 02, 2024. [Online]. Available: http://arxiv.org/abs/2406.07539

[3] https://realpython.com/python-gui-tkinter/#controlling-layout-with-geometry-managers

[4] https://jacobgil.github.io/deeplearning/vision-transformer-explainability#the-way-we-fuse-the-attention-heads-matters

[5] https://medium.com/the-dl/how-to-use-pytorch-hooks-5041d777f904

[6] https://github.com/openai/CLIP

[7] https://github.com/kzl/decision-transformer/blob/master/gym/experiment.py

[8] https://www.youtube.com/watch?v=A1tqsEkSoLg