Grounding Language in Motor Space: Exploring Robot Action Learning and Control from Proprioception

by

Emre Can Acikgoz

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

August 8, 2024

Grounding Language in Motor Space: Exploring Robot Action Learning and Control from Proprioception

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Emre Can Acikgoz

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Deniz Yuret (Advisor)

_____

Prof. Erkut Erdem

_____

Assist. Prof. Barış Akgün

Date: _____

To my grandmother...

ABSTRACT

Grounding Language in Motor Space: Exploring Robot Action Learning and Control from Proprioception

Emre Can Acikgoz

Master of Science in Computer Science and Engineering

August 8, 2024

Language development, particularly in its early stages, is deeply correlated with sensory-motor experiences. For instance, babies develop progressively via unsupervised exploration and incremental learning, such as labeling the action of "walking" by first discovering to move their legs via trial and error. Drawing inspiration from this developmental process, our study explores robot action learning by trying to map linguistic meaning onto non-linguistic experiences in autonomous agents, specifically for a 7-DoF robot arm. While current grounded language learning (GLL) in robotics emphasizes visual grounding, our focus is on grounding language in a robot's internal motor space. We investigate this through two key aspects: Robot Action Classification and Language-Guided Robot Control, both within a 'Blind Robot' scenario by relying solely on proprioceptive information without any visual input in pixel space. In Robot Action Classification, we enable robots to understand and categorize their actions using internal sensory data by leveraging Self-Supervised Learning (SSL) through pretraining an Action Decoder for better state representation. Our SSL-based approach significantly surpasses other baselines, particularly in scenarios with limited data. Conversely, Language-Guided Robot Control poses a greater challenge by requiring robots to follow natural language instructions, interpret linguistic commands, generate a sequence of actions, and continuously interact with the environment. To achieve that, we utilize another Action Decoder pretrained on sensory state data and then fine-tune it alongside a Large Language Model (LLM) for better linguistic reasoning abilities. This integration enables the robot arm to execute language-guided manipulation tasks in real time. We validated our approach using the popular CALVIN Benchmark, where our methodology based on SSL significantly outperformed traditional architectures, particularly in low-data scenarios on action classification. Moreover, in the instruction following tasks, our

Action Decoder-based framework achieved on-par results with large Vision-Language Models (VLMs) in the CALVIN table-top environment. Our results underscore the importance of robust state representations and the potential of the robot's internal motor space for learning embodied tasks.

ÖZETÇE

Dil Öğrenimini Robot Motor Alanında Temellendirme:
Propriyosepsiyondan Robot Eylem Öğrenimi ve Kontrolünü Keşfetmek
Emre Can Acikgoz
Bilgisayar Bilimler ve Mühendisliği, Yüksek Lisans
August 8, 2024

Dil gelişimi, özellikle erken evrelerinde, duyusal-motor deneyimlerle derinden ilişkilidir. Örneğin, bebekler denetimsiz keşif ve aşamalı öğrenme yoluyla aşamalı olarak gelişir; ilk önce deneme yanılma yoluyla bacaklarını hareket ettirmeyi keşfederek "yürüme" eylemini etiketlemek gibi. Bu gelişim sürecinden ilham alan çalışmamız, özellikle 7 serbestlik dereceli bir robot kolu için otonom robotlarda dilsel anlamı dilsel olmayan deneyimlerle eşleştirmeye çalışarak robot eylem öğrenimini araştırmaktadır. Robotik alandaki mevcut Temellendirilmiş Dil Öğrenimi (TDÖ) genelde görsel temellendirmeyi vurgularken, bizim odak noktamız dili bir robotun iç motor alanında temellendirmektir. Bunu iki temel açıdan araştırıyoruz: Robot Eylem Sınıflandırması ve Dil Kılavuzlu Robot Kontrolü, her ikisi de piksel uzayında herhangi bir görsel girdi olmadan yalnızca propriyoseptif bilgilere dayanarak bir 'Kör Robot' senaryosu içinde inceleniyor. Robot Eylem Sınıflandırmasında, daha iyi durum temsili için bir Eylem Çözücünün ön-eğitimi yoluyla Kendi Kendine Denetimli Öğrenmeden (KKDÖ) yararlanarak robotların duyusal verilerini kullanarak eylemlerini anlamalarını ve kategorize etmelerini sağlıyoruz. KKDÖ tabanlı yaklaşımımız, özellikle sınırlı veriye sahip senaryolarda diğer temel kalıpları önemli ölçüde aşmaktadır. Buna karşılık, Dil Güdümlü Robot Kontrolü, robotların doğal dil talimatlarını takip etmesini, dilsel komutları yorumlamasını, bir dizi eylemi oluşturmasını ve çevre ile sürekli etkileşimde bulunmasını gerektirerek daha büyük bir zorluk teşkil etmektedir.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Deniz Yuret, who has been my mentor since my undergraduate years. His guidance has consistently encouraged me to delve into the underlying reasons for my results, whether they were successful or not. Through his mentorship, I have learned the crucial skill of becoming an independent researcher, mastering the art of debugging and refining my own research. Most importantly, he has shown me how to find joy in the process, making the journey both enlightening and enjoyable.

I want to thank my Co-Advisor, Aykut Erdem, whose dedication to his work has been an inspiring example throughout my academic journey. He consistently encouraged my ambitious research goals, offering both the freedom to explore and the guidance to achieve them. His mentorship has been, always, truly invaluable.

I would also like to thank Erkut Erdem, who, despite the distance, has always been there during critical times; I will always miss his to-the-point paper suggestions. Yet another special thanks to Barış Akgün for his patience in introducing robotics to an NLP guy. I am grateful to Sajit Rao for his insightful discussions on embodied intelligence. I would like to thank Ekin Akyürek and Burak Soner for all of their support during my MSc period.

I extend my thanks to the KUIS AI Center, including all my professors, the technical staff, and my colleagues. Our fostered family-like environment has proven the value of "feeling at home" in one's professional life.

Heartfelt thanks to Jasmine Alara Tatari for her unwavering presence, silent support of my unconventional ideas, and acceptance of my stubborn realism. I want to express my deepest gratitude to Banu Açıkgöz and Harun Açıkgöz for their support. Thank you... Last but not least, I would like to thank Türkan Açıkgöz for encouraging me to always pursue my dreams.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CNN	Convolutional Neural Network
DL	Deep Learning
DoF	Degree of Freedom
EEF	End-Effector
FC	Fully Connected Network
GLL	Grounded Language Learning
GRU	Gated Recurrent Units
KL	Kullback-Leibler
LH-MTLC	Long-Horizon Multi-Task Language Control
LLM	Large Language Model
LSTM	Long Short-Term Memory
MCIL	Multi-context Imitation Learning
ML	Machine Learning
MLP	Multi-Layer Perceptrons
MSE	Mean Square Error
MTLC	Multi-Task Language Control
NLP	Natural Language Processing
RNN	Recurrent Neural Network
RO	Robot Observation
RSO	Robot and State Observation
SO	State Observation
SSL	Self-Supervised Learning
VLM	Vision-Language Model
VAE	Variational Auto-Encoder

Chapter 1

INTRODUCTION

1.1 Motivation

Language, especially in its early stages, is deeply correlated with sensory-motor experiences. Babies, for instance, learn actions and later associate these actions with language as linguistic labels, such as identifying the act of pushing or rotating something [Smith and Gasser, 2005]. These points highlight that intelligence is not merely embodied but becomes embodied through an **unsupervised** and **developmental** process. This progressive trajectory forms the foundation of a person's cognitive capabilities, emphasizing the importance of grounding language in physical experiences [Saffran et al., 1999, Hollich et al., 2000, Samuelson et al., 2011].

Grounded Language Learning (GLL) aims to map the meaning of language with non-linguistic experiences in autonomous agents, such as robots. These agents can also be termed as grounded due to addressing the symbol grounding problem [Harnad, 1990]. This domain includes systems that translate language into physical actions, make linguistic decisions using non-linguistic input, and learn how to use language in a grounded setting.

Current approaches in GLL in the robotic field have predominantly focused on visual grounding which leaves a significant gap in how language can be grounded in motor actions. For example, the phrase "rotate right" can be mapped to a specific set of joint velocities of the end-effector (EEF) of the robot given a particular joint configuration of the robot arm. This allows us to label motions and robotic tasks with language and to follow instructions such as "rotate the blue block right."

This thesis addresses the question of *"Can we learn good and accurate representations from an agent's internal motor space?"*. We explore this by investigating

language grounding within a robot's motor space, a crucial yet underexplored area of GLL. To demonstrate our approach, we applied our hypothesis to two distinct robotic downstream tasks: **Robot Action Classification** and **Robot Instruction Following**. Our method achieved state-of-the-art results in Robot Action Classification and competitive performance with vision-language models (VLMs) in Instruction Following.

1.2 Background

1.2.1 Evolution of Deep Learning Architectures

The evolution of Deep Learning (DL) architectures has played a crucial role in advancing AI. Starting with Multi-Layer Perceptrons (MLPs) [Rosenblatt, 1958a], which provided a foundation for neural network-based solutions, the field progressed to Recurrent Neural Networks (RNNs) that could process sequential data [Hochreiter and Schmidhuber, 1997, Sutskever et al., 2014a, Sarzynska-Wawer et al., 2021], mirroring aspects of temporal cognition. Convolutional Neural Networks (CNNs) then revolutionized visual processing, analogous to the visual cortex in human brains, and advanced the field of computer vision [Lecun et al., 1998, Krizhevsky et al., 2012, Dosovitskiy et al., 2020]. The introduction of Transformers marked a significant leap by enabling models to handle long-range dependencies in data, particularly beneficial for language-oriented tasks [Vaswani et al., 2023].

1.2.2 The Era of Large Language Models

The transformer mechanism revolutionized natural language processing (NLP) by introducing a self-attention mechanism and the support of parallelism [Vaswani et al., 2023]. This architecture paved the way for Self-Supervised Learning (SSL) in NLP, where models are pre-trained on vast amounts of unlabeled text data as a pretext task and then fine-tuned on specific downstream tasks [Devlin et al., 2018, Radford et al., 2019a] such as summarization, topic classification, sentiment analysis. Through SSL, one can leverage unlabeled data to learn general representations. It leads the model to capture inherent data structures and patterns that can then be

fine-tuned efficiently for specific downstream tasks with a limited amount of labeled data. On the other hand, the rapid advancement in GPU technology, coupled with the availability of large-scale datasets and the scaling of model parameters, has enabled the training of increasingly powerful language models and achieved significant results which led to the development of Large Language Models (LLMs) [Radford et al., 2019a, Devlin et al., 2018, Radford et al., 2019b, Brown et al., 2020, Chowdhery et al., 2023, Touvron et al., 2023, Achiam et al., 2023, Team et al., 2023].

The latest frontier, multimodal learning, represents a closer approximation to human-like intelligence by integrating multiple sensory of inputs, such as vision and language [Vinyals et al., 2015, Perez et al., 2018, Alayrac et al., 2022]. The progression towards multimodal LLMs aligns closely with our understanding of how human intelligence since the idea of "multimodal" learning develops through the integration of various sensory representations and interactions with the environment benefiting from the scale and reasoning abilities of the LLMs [Alayrac et al., 2022, Awadalla et al., 2023, Li et al., 2023a, Liu et al., 2024]. Each step in this evolution has brought AI systems closer to the kind of embodied, contextual, and multimodal intelligence that characterizes human cognition.

1.2.3 Grounding Language with Robot Learning

Significant progress has been made in robotic learning and manipulation in recent years, particularly in language-conditioned instruction following tasks. Inspired by this incremental and multimodal learning[1], the field of GLL has emerged as a promising approach to enhance the capabilities of autonomous robots, in understanding and using natural language within their world. This progress is attributed to four major approaches: (1) building sophisticated architectures from the ground up to combine language policies with visual perception [Lynch and Sermanet, 2020, Brohan et al., 2022], (2) creating efficient fine-tuning methods for better modality representations

[1]Multimodal learning integrates information from multiple types of data modalities, such as visual, text, and tactile inputs, allowing for a more comprehensive and integrated understanding of the world.

[Mees et al., 2022b, Mees et al., 2022a], (3) utilizing LLMs as powerful planners and reasoner for multi-task instruction solving [Ahn et al., 2022], and (4) merging Vision-Language Models (VLMs) as high-level planners with policy heads for direct low-level control outputs [Li et al., 2023b]. These methodologies have collectively improved robots' capability to comprehend and carry out language-based commands in complex settings and environments.

Despite the promise of GLL, the majority of existing approaches have predominantly focused on grounding language in a visual space [Alayrac et al., 2022, Mees et al., 2022b, Mees et al., 2022a, Brohan et al., 2022, Li et al., 2023b]. There is a notable scarcity of studies exploring grounding within "motor" space, and even fewer that address a visuo-motor space. We aim to fill this gap by experimenting with language grounding in a robot's internal space.

1.3 Scope of Thesis

Taking these into consideration, this thesis explores the question *"Can we learn good and accurate representations from an agent's internal motor space?"* by integrating robot proprioceptive information with NLP for Robot Action Classification and Robot Instruction Following, without using vision as perceptual input.

Robot Action Classification. Action classification is a fundamental task in AI and Machine Learning (ML), aimed at identifying and categorizing specific activities or behaviors from the observed data. This capability is crucial across various domains, from human activity recognition from videos to gesture-based interfaces in human-computer interaction. In robotics, action classification takes on a unique and challenging dimension. In our case, Robot Action Classification specifically focuses on robot manipulation tasks in a tabletop environment, testing if a robot understands and categorizes its actions and behaviors. Motivated by our thesis question, we aim to achieve this understanding using only proprioceptive information - the robot's internal sense of position, movement, and states. This approach simulates a "blind robot" scenario and pushes the boundaries of robot perception and cogni-

tion. Our research leverages recent advancements in transformer architectures and LLMs to tackle this challenge by employing novel techniques such as self-supervised pre-training and curriculum learning to improve classification accuracy, especially in low-data scenarios and few-shot settings.

Robot Instruction Following. Robot Instruction Following is an advanced frontier in robotics, where natural language instructions are used to direct a robot's actions in completing complex human-driven tasks. In our research, we explore this challenge within the unique constraints of a *blind robot* setting, relying solely on proprioceptive information without visual input as it is in Robot Action Classification. This approach significantly increases the difficulty of the task, as the robot must not only understand and interpret linguistic commands but also generate appropriate action sequences and interact successfully with its environment without any visual feedback from pixels. To address these challenges, we leverage the power of LLMs with multimodal fusion strategies from Vision-Language Models (VLMs). Here LLMs serve as a bridge between human language and robot actions, enabling the system to break down high-level tasks into actionable steps. The robot must then execute these steps through a series of generated actions or "State Trajectories"[2] constantly adjusting its behavior based on its internal sensory feedback and the proceeding states.

1.4 Thesis Contributions

We evaluate our approaches using CALVIN [Mees et al., 2022b], an open-source simulated benchmark designed for long-horizon, language-conditioned tasks. This benchmark includes 34 distinct tasks for robot action classification and assesses 1000 unique instruction chains for sequential tasks in language-guided robot manipulation, where the robot must successfully complete sequences of up to five consecutive language instructions. Our contributions are as follows:

[2]The state trajectory refers to the sequence of state snapshots captured during the rollout as the robot executes the given instruction.

1. Representing the meaning of words within the robot's internal motion space.

2. Generating robot actions from language and categorizing tasks based on the robot's internal actions.

3. Demonstrating the connection between language and robot proprioceptive information, akin to vision-based perception, but without using pixels.

4. Introducing the first attempt at SSL solely on robot motion space.

5. Showing the potential of robot abstract state representations in embodied task learning.

This work not only enhances robot self-awareness and adaptability but also opens up new avenues for grounding language into state motion space.

1.5 Thesis Structure

The structure of this thesis is organized as follows:

- **Chapter 2** delves into the relationship between language and sensory-motor experiences, emphasizing the role of SSL in robot action classification. It introduces the CALVIN dataset, describing the environment, dataset collection process, challenges faced, and sensor features. The chapter also details the model architecture, including input features, the self-supervised approach, and probing techniques. The experiments and results section covers the Calvin benchmark, robot manipulation tasks, baselines, evaluation methods, robot action classification, data scaling, and qualitative analysis. The chapter concludes by summarizing the key findings, highlighting the innovative use of SSL with solely sensorimotor inputs.

- **Chapter 3** addresses the challenges and significance of directing robot actions through natural language instructions, especially within a "blind robot" setting that relies exclusively on proprioceptive information. It reviews related

work, including studies on MCIL, HULC, RT-1, Flamingo, and RoboFlamingo. The chapter describes the CALVIN environment setup for language-guided tasks and explains the pre-training of the State Embedder and fine-tuning of Blind-RoboFlamingo, integrating language representations with robot motor space. The experiments and results section presents benchmarks, baselines, results, and the impact of different proprioceptive encoders on model performance. The chapter concludes by summarizing the integration of language and sensory-motor experiences in the "blind robot" setting and its implications for robotics.

- **Chapter 4** provides a comprehensive summary of the entire thesis, highlighting the main contributions and findings. It also discusses potential future directions and improvements based on the current research, offering insights into the continued development and enhancement of robotic systems.

Chapter 2

ROBOT ACTION CLASSIFICATION

2.1 Introduction

Researchers in cognitive science and AI have long been fascinated by the intricate relationship between language and sensory-motor experiences. Language development, especially in its early stages, is deeply intertwined with sensory-motor capabilities and experiences in babies [Smith and Gasser, 2005]. For example, a baby might put its thumb in its mouth or kick their legs inside the mother's womb during pregnancy. Moreover, a newborn first learns to walk and, in subsequent developmental stages, identifies this process as "walking" by recognizing the action of moving its legs. This progression illustrates how motor actions are learned first and then categorized with linguistic action labels, supporting the idea that a baby's learning is unsupervised and incremental.

Recent advancements in self-supervised learning (SSL) have shown remarkable potential in various domains such as text [Devlin et al., 2018, Radford et al., 2019a] and vision [Dosovitskiy et al., 2020]. However, the application of this method within the robot proprioceptive space, specifically for action modality, remains largely unexplored. This gap presents a significant opportunity to enhance our understanding and capabilities in robot action classification, a critical aspect for the development of more autonomous and adaptive robotic systems.

In this part of our study, we introduce a novel approach that leverages SSL to train action representations in the robot proprioceptive space. Inspired by the analogy to baby learning discussed above, where learning is unsupervised and incremental, we first develop an unsupervised representation of primitive actions performed by a robot arm in a tabletop environment. These learned representations are then utilized in downstream action classification tasks. This is achieved through autore-

gressive training, followed by freezing the State Embedder to probe it for action classification. Our methodology incorporates curriculum learning, which has been shown to significantly outperform traditional from-scratch training approaches, particularly in low-data scenarios. This aspect is crucial as it addresses one of the fundamental challenges in robotics: effective learning from limited data.

Our methodology for SSL is a two-step process: pre-training on robot observation followed by a stage of transfer learning (Figure 2.1). In the pre-training phase, we employ autoregressive training on the robot's proprioceptive states, the delta actions (i.e., relative actions), and the difference of the scene observations (Figure 2.2). This phase focuses on capturing the intrinsic relationships and dynamics within the robot's observations and actions, enabling the model to learn robust and generalized representations. Upon completion of the pre-training phase, we transfer the pre-trained representations to the downstream task of action classification (Figure 2.3). This transfer learning stage allows us to leverage the rich feature representations learned during pre-training, enhancing the performance and accuracy of the action classification task.

To further validate our approach, we employ t-SNE [van der Maaten and Hinton, 2008] for enhanced visual analysis. This technique allows us to observe the clustering of related actions within the embedding space, thus demonstrating the efficacy of the latent embeddings learned through self-supervision. The State Embedder's ability to effectively cluster these actions underscores the robustness of our methodology and the quality of the learned representations.

Our research bridges the gap between an agent's intricate motor skills and its actions, providing deeper insights into the action-learning processes in robotics. By doing so, we enhance our grasp of how robots can learn and adapt their actions more efficiently. The findings and methodologies presented in this thesis contribute to the broader field of robotics, offering a pathway to more advanced and adaptive robotic systems. Moreover, all the source code developed during this research is publicly available, promoting transparency and enabling further research and development

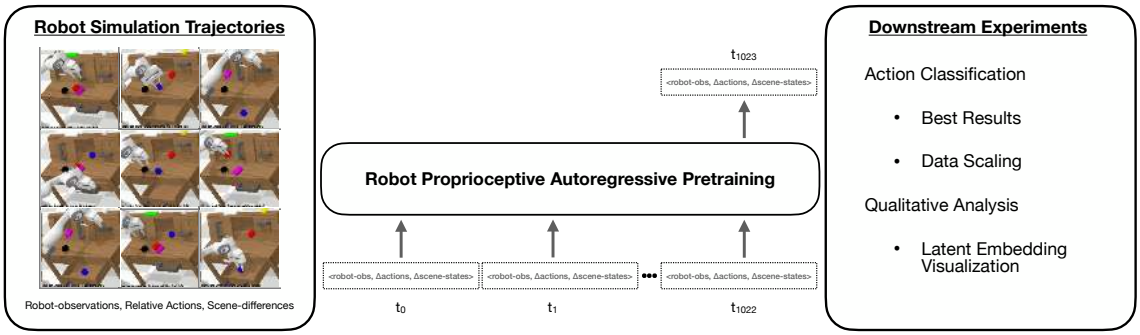

Figure 2.1: **Action learning with robot proprioceptive pre-training.** *Left:* We utilized the CALVIN Benchmark Dataset, which comprises proprioceptive robot observations, scene observations, and actions, for specialized pre-training focused on proprioceptive actions. *Middle:* By treating trajectory observations similarly to the next token prediction in language models, we trained a decoder model to forecast future states based on the preceding context. *Right:* This pre-training enabled the transfer of these learned representations to a variety of downstream tasks and experiments.

in this field [1].

2.2 CALVIN Dataset

During our training and experiments, we used the CALVIN (Composing Actions from Language and Vision) [Mees et al., 2022b] which was originally designed to benchmark language-conditioned policy learning for long-horizon robot manipulation tasks. Thanks to its rich sensory-motor of observations and complex sequences of action tasks, CALVIN supports the development of agents capable of solving complex robotic manipulation tasks specified solely through human language instructions. This benchmark spans different but structurally related manipulation environments in table-top settings, providing a rich set of multimodal observations to facilitate training and evaluation of advanced robotic agents.

2.2.1 Environment

CALVIN includes four different environments (A, B, C, D) designed to test the generalization and skill acquisition of robot policies. Each environment includes a

[1] `https://github.com/emrecanacikgoz/goal-classifier`

7-DOF Franka Emika Panda robot arm equipped with a parallel gripper and some interactive elements such as a sliding door, a drawer, buttons, and switches. The environments differ in textures and positions of static elements to challenge the robot's ability to generalize learned tasks. Physics simulation is powered by the PyBullet[2] engine, ensuring realistic interactions and efficient data collection. The collected CALVIN dataset supports a variety of sensors, including RGB-D cameras (both static and gripper-mounted), proprioceptive sensors, and vision-based tactile sensors. The robot operates in a closed-loop continuous control setting by sending actions at a frequency of 30 Hz. It supports experimentation with both absolute and relative Cartesian action spaces. In the dataset, there 34 specific tasks across the four environments.

2.2.2 Dataset Collection

The dataset is comprised of unstructured teleoperated demonstrations and corresponding language instructions that provide a comprehensive resource for training and evaluating robotic agents. Twenty-four hours of teleoperated play data were collected using an HTC Vive VR headset, resulting in approximately 2.4 million interaction steps and 40 million short-horizon windows for relabeled goal-conditioned imitation learning. The play data, collected by three untrained operators, is diverse and task-agnostic, covering a wide range of exploratory and sub-optimal behaviors critical for learning robust and generalizable representations. On the other hand, the dataset includes over 20,000 crowd-sourced natural language directives corresponding to the 34 defined tasks. Only 1% of the robot interaction data is annotated with language instructions, simulating real-world scenarios where exhaustive annotations are impractical.

2.2.3 Challange

The CALVIN benchmark presents a series of evaluation protocols of varying difficulty to test the robustness and generalization capabilities of trained agents. Three

[2] `https://github.com/bulletphysics/bullet3`

combinations of training and test environments are provided: Single Environment (train on D and evaluated D), Multi Environment (train on ABCD and evaluated D), and Zero-Shot Multi Environment (train on ABC and evaluated D). In the Single environment setup, training and evaluation occur in the same environment. This setup allows the agent to become highly specialized in one environment, making it easier to assess the baseline performance and task completion capabilities in a controlled setting. In Multi Environment setup, training spans all four environments, and evaluation is conducted in one of them. This setup introduces variability in textures and locations of elements such as the sliding door, button, and switch, challenging the agent's ability to generalize learned skills across different but related environments. Finally, Zero-Shot Multi Environment training covers three environments, while evaluation is conducted in an unseen fourth environment. This is the most challenging scenario, as the agent must generalize its learned skills to a novel setting with different textures and element positions that it has never encountered during training.

The CALVIN benchmark uses two primary evaluation metrics to assess the performance of robotic agents:

- Multi-Task Language Control (MTLC): This metric evaluates the agent's ability to perform 34 distinct manipulation tasks specified through language instructions. For each task, the environment is reset to the initial state of a valid unseen demonstration to ensure the commanded instruction is feasible. The agent must complete 10 rollouts for each task from different starting states, with the language instructions used for testing being novel and not included in the training set. The success rate of task completion is recorded, providing a measure of the agent's ability to understand and execute individual language directives.

- Long-Horizon Multi-Task Language Control (LH-MTLC): This evaluation extends the MTLC metric by testing the agent's capability to accomplish sequences of multiple language instructions in a row. Specifically, the agent must complete five sequential tasks, forming a long-horizon task chain. The

evaluation sequences are curated to exclude infeasible, cyclic, or overly similar tasks, resulting in 1000 unique instruction chains. Each chain requires the agent to transition between different subgoals, testing its ability to maintain context and continuity across multiple instructions. The robot is reset to a neutral position after each sequence to prevent biases from initial poses, and various initial scene configurations are included to further evaluate generalization.

2.2.4 Sensor Features

CALVIN supports various combinations of sensor inputs, encouraging the exploration of different perceptual strategies to tackle real-world robotic manipulation challenges. The sensors include RGB-D images provided by both static and gripper-mounted cameras, offering a comprehensive visual representation of the environment from different perspectives; proprioceptive information, which is internal sensor data from the robot capturing joint positions, velocities, and forces; and vision-based tactile sensors, which provide detailed tactile feedback useful for tasks requiring fine-grained manipulation, such as maintaining a stable grasp on objects. These sensor combinations are evaluated to understand their impact on the agent's performance in both single and long-horizon tasks, providing insights into the most effective sensor suites for different types of robotic manipulation challenges.

In summary, the CALVIN benchmark offers a robust and flexible framework for evaluating language-conditioned robotic agents. By incorporating diverse environments, detailed evaluation metrics, and various sensor combinations, CALVIN aims to drive advancements in scalable, general-purpose, language-driven robotics.

2.3 Model

2.3.1 Input Features.

Our inputs comprise:

- robot observations as (x, y, z) world coordinates and (e_x, e_y, e_z) euler angles

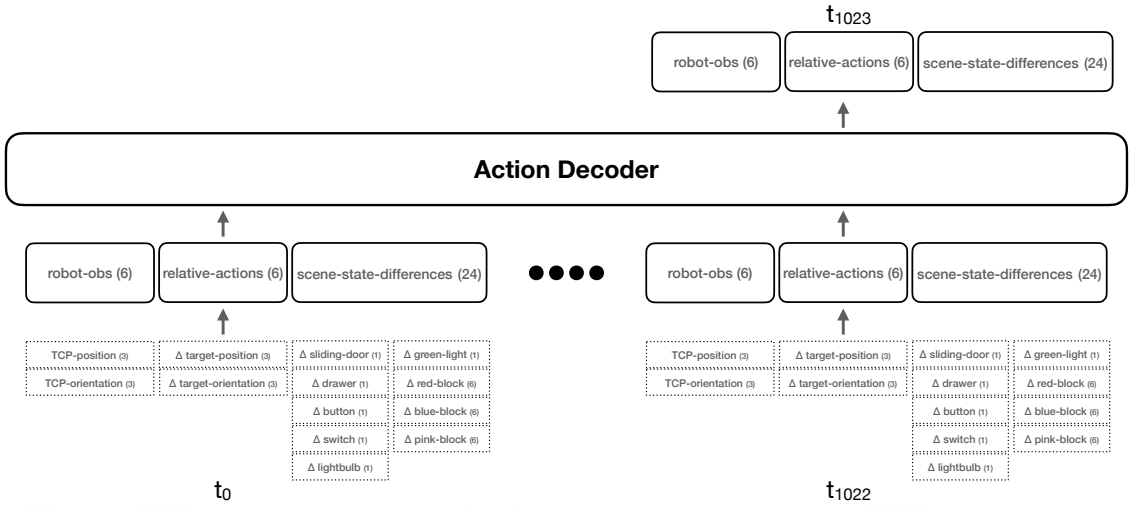

Figure 2.2: **Proprioceptive Autoregressive pre-training.** Our model functions as a decoder, processing sequences of 1024-window observations from the robot arm, along with the differential scene observations at each time step. Instead of encoding these sensorimotor inputs, we directly employ their continuous values as embeddings. The model is trained using an autoregressive learning approach for next-state prediction, where the decoder is specifically trained to anticipate the subsequent state based on preceding states.

of the tool-center-point,

- $(\Delta x,\ \Delta y,\ \Delta z)$ relative world coordinates and $(\Delta e_x,\ \Delta e_y,\ \Delta e_z)$ relative euler angles,

- Scene state differences (Δ-scene-observations) of sliding door, drawer, button, switch, lightbulb, green light, red block, blue block, and pink block between each frame.

Robot observations are in cartesian space and define the desired tool-center-point pose with respect to the world frame. Tool-center-point is a virtual frame between the gripper fingertips of the robot. The scene states in the CALVIN environment capture 24 distinct variables, encompassing the position and orientation of all scene objects. These include the joint state of the sliding door, drawer, button, and switch, as well as binary values indicating the on/off status of the lightbulb and green light. Additionally, the coordinates (x, y, z) and Euler angles (euler-x, euler-y, euler-z) of the red, blue, and pink blocks are recorded. We process the delta changes in scene

observations at each timestep as *scene-differences*. During our training, we didn't use any image data as perception and learned everything from the robot's internal motorspace together with the delta state observations.

Scene differences. To emphasize only the incremental changes of target objects, we implement a data augmentation strategy by extracting and normalizing the differences in scene coordinates. Starting from the initial scene information at s_0, we calculate the difference for subsequent timesteps as $diff_i = scene_i - scene_{i-1}$.

2.3.2 Architecture

For the model's input, we utilize a 36-dimensional tensor composed of floating-point numbers. This tensor is mapped to the model's embedding space through a linear, fully-connected layer. We refer to this layer as *FC-in*, functioning as a learnable token embedder for the continuous robot and scene states. The mapped states passed to a decoder that follows the GPT-2 [Radford et al., 2019b] architecture, incorporating 8 decoder layers and 8 attention heads. The model's embedding size is configured at 512, and it employs GeLU [Hendrycks and Gimpel, 2016] for non-linear activation functions. We utilize CNN-layers in the feedforward blocks as a replacement for MLPs. This architecture enables the model to predict latent representations for each rollout window.

Our model is a decoder-only Transformer with 1M parameters. It comprises eight transformer blocks, each equipped with eight attention heads. The hidden dimension of the decoder is 512, and it uses a CNN ratio of 4. The model processes inputs through 1024 windows, with each window capturing sensorimotor features derived from robot observations including proprioceptive states and changes in the scene. To accommodate the varying dimensions of different inputs (such as 15-dimensional arrays for robot observations and 24-dimensional arrays for scene differences), we used an input linear layer to project each modality into the dimension of 512.

2.3.3 Self-Supervised Approach

In our self-supervised representation learning method, we concentrate on predicting the next timestep of observations using a State Embedder. This involves analyzing sequences of snapshots captured at each time step (t) within a specified time-window (T), which reflects both the robot and its surroundings. Our approach is based on an autoregressive objective, which is designed to compute the likelihood $P(S)$ of a series of states $S = (s_0, s_1, ..., s_T)$. $P(S)$ is expressed as the product of conditional probabilities:

$$P(S) = \prod_{t=0}^{T} P(s_t|s_{<t}) \tag{2.1}$$

Essentially, our goal is to accurately predict the likelihood of a sequence of states and actions. To summarize, our primary aim is to forecast the robot's next state of information based on prior context via the next-token-prediction objective by masking the future tokens.

Autoregressive pre-training. The data is segmented into 1024 windows, and following this, the decoder is trained in an autoregressive fashion as in language models [Radford et al., 2019b]. This training approach aims to generate next-state features by utilizing the given prior context within the fixed 1024 window size. At each time step within the window, the loss is calculated by comparing the continuous ground truth states and the generated states, using Mean Square Error (MSE) as the optimization criterion. In the sensorimotor pre-training phase, we utilize fixed-sized windows of 1024 to train the decoder model. This training involves predicting subsequent window based on preceding action windows. Additionally, we explore various window sizes, including [256, 512, 1024], as detailed in our ablation studies in Section X. We employ an autoregressive MSE loss, uniformly weighting each feature in the input. The models are trained for 300 epochs, with a batch size of 32, using the AdamW optimizer. The optimizer settings include a learning rate of 4×10^{-4} and a weight decay of 0.01.

2.3.4 Probing

Probing in deep learning is employed to decipher the hidden features and knowledge encoded by a model, using linearly projected classifiers [Belinkov, 2021]. To understand the nature of the hidden representations our State Embedder has learned about the robot's internal sensorimotor actions and the given task, we extract the final output of the State Embedder, with dimensions [B, T, E]. We then apply max-pooling, capturing the most activated neurons across the time dimension, resulting in a reshaped output of [B, 1, E]. Additionally, we explored various pooling techniques, including max-pooling, average-pooling, and final time-step pooling, as part of our ablation studies. These processed embeddings are then projected into a 34-class space for action classification experiments.

2.4 Experiments & Results

2.4.1 Calvin Benchmark and Robot Manipulation Tasks

Calvin Benchmark. We used CALVIN [Mees et al., 2022b] as our source dataset which is an open-source benchmark for teaching agents to perform long-horizon, language-driven robotic tasks. It focuses on complex sequences, diverse actions, and language interpretation, using onboard sensors and flexible sensor suites. CALVIN consists of four structurally similar yet distinct environments (A, B, C, D) for general play which is based on PyBullet physics engine. Each environment features a robot arm with a parallel gripper, a desk with a sliding door and drawer, a button toggling a green light, and a light bulb switch. Additionally, there are three uniquely colored and shaped rectangular blocks. To assess language grounding generalization, each environment varies in textures and positions of static elements like doors, drawers, and switches. The desk, robot, and static camera placement remain constant across environments. We train the models in A, B, C, and D environments together using the training data and evaluate them in the D environment using the unseen validation data.

Robot and Tasks. In the CALVIN environment, the play data is generated by a human operator using a 7-DoF Franka Emika Panda robot equipped with a 1-DoF parallel jaw gripper. This robot operates in a tabletop setting, where it captures snapshots from simulations conducted at a frequency of 30 Hz, utilizing joint position control. The CALVIN benchmark comprises 34 distinct tasks, including pushing, rotating, and lifting objects, all of which we also employ in our study. Detailed descriptions of these tasks are provided in Figure 2.6.

2.4.2 Baselines

For our baselines, we begin with fundamental deep learning architectures, employing vanilla MLPs with three layers, along with versions of RNN, LSTM [Hochreiter and Schmidhuber, 1997], GRU [Chung et al., 2014], and a Transformer [Vaswani et al., 2023], all trained from scratch. We provide an overview of these architectures by providing the mathematical formulations and key advancements.

Multi-Layer Perceptrons (MLPs). MLPs are the foundational architecture of deep learning, consisting of multiple layers of neurons with each layer fully connected to the next [Rosenblatt, 1958b, Rumelhart et al., 1986, Hornik et al., 1989, Bishop, 1995, LeCun et al., 1998]. Each neuron in an MLP applies a weighted sum of inputs followed by a non-linear activation function. Mathematically, the output $\mathbf{y}$ of an MLP with one hidden layer can be expressed as:

$$\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \tag{2.2}$$

$$\mathbf{y} = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2 \tag{2.3}$$

where $\mathbf{x}$ is the input vector, $\mathbf{W}_1$ and $\mathbf{W}_2$ are weight matrices, $\mathbf{b}_1$ and $\mathbf{b}_2$ are bias vectors, and σ is the activation function, typically a ReLU [Agarap, 2019] or sigmoid function. Despite their simplicity, MLPs are powerful function approximators and have been successfully applied to a variety of tasks, such as image and speech recognition, when used as building blocks in more complex architectures.

Recurrent Neural Networks (RNNs). RNNs are designed to handle sequential data by maintaining hidden states that capture temporal dependencies [Elman, 1990, Graves, 2012]. The fundamental unit of an RNN is the recurrent cell, which updates its hidden state $\mathbf{h}_t$ at time step t based on the input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. This update can be described by the following equations:

$$\mathbf{h}_t = \sigma(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h \mathbf{h}_{t-1} + \mathbf{b}_h) \tag{2.4}$$

$$\mathbf{y}_t = \mathbf{V}_h \mathbf{h}_t + \mathbf{c}_h \tag{2.5}$$

where σ is the activation function, $\mathbf{W}_h$, $\mathbf{U}_h$, and $\mathbf{V}_h$ are weight matrices, and $\mathbf{b}_h$ and $\mathbf{c}_h$ are bias vectors. Despite their potential, vanilla RNNs suffer from vanishing and exploding gradient problems, which hinder learning long-term dependencies [Bengio et al., 1994, Pascanu et al., 2013].

Long Short-Term Memory (LSTM). LSTM are a specialized type of RNN designed to overcome the limitations of standard RNNs by incorporating memory cells that can maintain information over long periods, i.e. vanishing gradient problem [Hochreiter and Schmidhuber, 1997]. An LSTM cell contains three gates: input gate, forget gate, and output gate, which regulate the flow of information. The equations governing an LSTM cell are as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{2.6}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{2.7}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \tag{2.8}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \tag{2.9}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{2.10}$$

where $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$ are the input, forget, and output gates, respectively, $\mathbf{c}_t$ is the cell state, and $\odot$ denotes element-wise multiplication. LSTMs had been widely used in tasks requiring long-term dependency learning, such as language modeling and speech recognition [Sutskever et al., 2014b, Cho et al., 2014].

Gated Recurrent Units (GRUs). GRUs are a simplified variant of LSTMs that combine the input and forget gates into a single update gate [Chung et al., 2014]. The GRU update equations are:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_z) \tag{2.11}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1} + \mathbf{b}_r) \tag{2.12}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h) \tag{2.13}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \tag{2.14}$$

where $\mathbf{z}_t$ is the update gate, $\mathbf{r}_t$ is the reset gate, and $\tilde{\mathbf{h}}_t$ is the candidate hidden state. GRUs are computationally efficient and have been shown to perform comparably to LSTMs on various tasks.

Transformers. Transformer module is a groundbreaking architecture that relies on self-attention mechanisms rather than recurrence to model sequential data [Vaswani et al., 2023]. The self-attention mechanism computes attention scores to weigh the importance of different tokens in a sequence. The self-attention operation is defined as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{2.15}$$

where $\mathbf{Q}$ (queries), $\mathbf{K}$ (keys), and $\mathbf{V}$ (values) are input matrices, and d_k is the dimension of the keys. The Transformer architecture consists of an encoder-decoder structure with stacked self-attention layers and position-wise feedforward layers. Each encoder and decoder layer includes multi-head attention and layer normalization. The Transformer's ability to capture long-range dependencies and parallelize training has led to significant advancements in NLP tasks. Notable Transformer models, such as BERT [Devlin et al., 2018], GPT [Radford et al., 2019a, Radford et al., 2019b, Brown et al., 2020, Achiam et al., 2023], and T5 [Raffel et al., 2019], have achieved state-of-the-art performance across various benchmarks.

We conducted an extensive grid search to optimize parameters such as hidden dimensions, number of layers, learning rate, dropout, and weight decay for MLP,

RNN, LSTM, and GRU during our experiments. Additionally, for the Transformer architecture, we explored the choice of feedforward network types, either MLP (similar to [Vaswani et al., 2023]) or CNN (as used in [Brown et al., 2020]), along with varying the number of attention heads.

2.4.3 Evaluation

To assess the effectiveness of our approach and benchmark it against baseline models, we conduct both quantitative and qualitative analyses in Section 2.4.6. For the quantitative aspect, we probe the final hidden layer of the State Embedder by training it for action classification alongside other baseline models. This examination will be conducted from distinct viewpoints: (i) the highest accuracy achieved (Section 2.4.4) and (ii) accuracy across a range of data scaling scenarios, from low to high (Section 2.2). Additionally, in terms of qualitative analysis, we will employ techniques like t-SNE [van der Maaten and Hinton, 2008] and scatter plots to visually represent and analyze the clustering patterns of each input (Section 2.4.6).

In our downstream experiments, we trained all baseline models from scratch, utilizing the same input features employed in pre-training the State Embedder: robot observations, relative actions, and scene differences. The model was trained using 1% of the labeled training data, which comprises 5,000 task-annotated frames with a window length of 64. For evaluation, we utilized the provided evaluation dataset by the CALVIN Benchmark [Mees et al., 2022b].

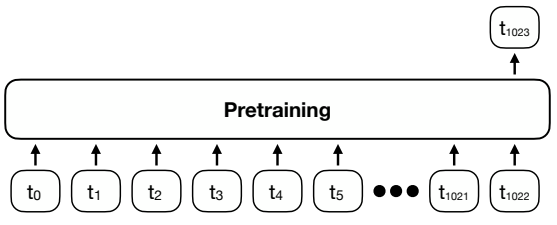

Figure 2.3: **Autoregressive pre-training Phase.** Step 1: Autoregressive pre-training phase of the Action-Decoder as a pretext task.

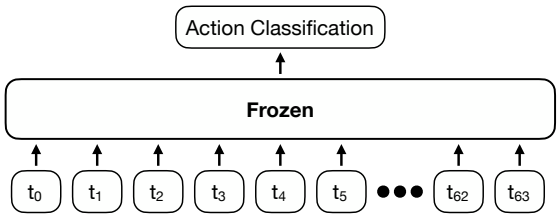

Figure 2.4: **Probing Phase for Action Classification.** Step 2: Probing the Action-Decoder for downstream action classification task.

Method	**Class Accuracy**
MLP	90.6
RNN	91.0
LSTM	91.1
GRU	<u>92.8</u>
Transformer	92.0
SSL + Probing (Ours)	**98.1**

Table 2.1: **Robot Action Classification Accuracy Reulsts.** Classification accuracies for CALVIN actions.

2.4.4 Robot Action Classification

The initial task involves action classification on CALVIN's evaluation dataset, which comprises 1,000 samples across 34 distinct labels (refer to Figure 4 for the labels). For this task, we selected MLP, RNN, LSTM, GRU, and Transformer as baseline models and trained them from scratch. To ensure reliable results, we conducted an extensive hyperparameter search, executing a minimum of 1,000 runs for each model under various hyperparameter configurations using Weights and Biases framework[3]. To compare these baselines with our action-decoder, we freeze the decoder and train one linear probing layer to the embeddings of the final hidden layer of the decoder (Figure 2.3).

Our fine-tuned State Embedder achieved state-of-the-art performance, reaching an accuracy of 98.1%, thereby surpassing the nearest baseline by 5.3%. This demonstrates that our curriculum learning-based, self-supervised approach is highly effective in capturing robust representations of knowledge for targeted actions, see Table 2.1 for the results.

[3] `https://wandb.ai`

2.4.5 Data Scaling

An effective and general artificial model should perform well in scenarios with limited data and be capable of learning from a few examples [Brown et al., 2020]. To assess our methods from this perspective, we conducted three distinct training experiments. Initially, we used only one random example ($N = 1$) from each of the 34 classes for training, relying on these 34 random samples for each class and evaluating performance on the full dataset. This process was repeated by training with two samples per class ($N = 2$, totaling 68 training samples) and subsequently with four samples per class ($N = 4$, totaling 136 training samples).

When we check Table 2.2, our pre-trained State Embedder achieved the highest performance, attaining an accuracy of 59.0% when trained with one sample per class ($N = 1$), significantly outperforming all other baselines. The nearest baseline, RNN, achieved a maximum accuracy of 28.7%, which is less than half of that attained by our State Embedder. Similarly, when we checked for two samples ($N = 2$) and four samples ($N = 4$) per class, our model consistently outperformed the other baselines by a significant margin. Table 2.2 demonstrates that our transfer learning approach is highly effective in the action domain, showcasing its capability to learn effectively from just a single sample.

We conducted experiments with various input settings, covering both the Robot Observations and the Scene Observations, to assess the impact of different input features on Robot Action classification. Due to the significant influence of scene differences compared to other input types, we focused on this aspect in our further analysis. The detailed results are presented in Appendix A.1.

2.4.6 Qualitative Analysis

To assess the efficacy of our decoder in clustering actions within its latent space, we analyze the final embeddings from the decoder's last layer. This is achieved by applying t-SNE [van der Maaten and Hinton, 2008] to these embeddings and visualizing the results in a two-dimensional space for qualitative analysis. As depicted

Method	**n=1**	**n=2**	**n=4**
MLP	24.2	44.8	65.2
RNN	<u>28.7</u>	54.0	66.1
LSTM	26.3	52.6	62.0
GRU	25.0	<u>59.4</u>	67.8
Transformer	22.8	46.7	<u>69.9</u>
SSL+Probing (Ours)	**59.9**	**65.9**	**77.8**

Table 2.2: **Data Scaling Experiments on Action Classification.** Classification accuracies for CALVIN actions with varying amounts of training data.

in Figure 2.5, it is evident that similar actions (indicated by identical colors) are predominantly clustered together, or at the very least, targets represented by the same symbols are in close proximity. This observation qualitatively substantiates the decoder's capability to effectively perform unsupervised clustering, relying solely on the robot's proprioceptive information or states.

2.5 Conclusion

In this part of our research, we pioneered the application of SSL, uniquely utilizing only the robot's sensorimotor inputs without relying on visual perception. Our approach treats the robot's proprioceptive information as a sequence modeling challenge, akin to language modeling. By pre-training a decoder in a huge unlabeled CALVIN play dataset, we learn to represent this information accurately, trainings with 30 minutes. The goal is to leverage this representation in downstream tasks by achieving both efficiency and accuracy. Our approach significantly surpasses other well-established baselines, particularly in scenarios with limited data. It demonstrates a remarkable capability to learn effective representations from just a few samples. Our work successfully connects the agent's complex motor skills with its actions, thereby enriching our understanding of the action-learning process.

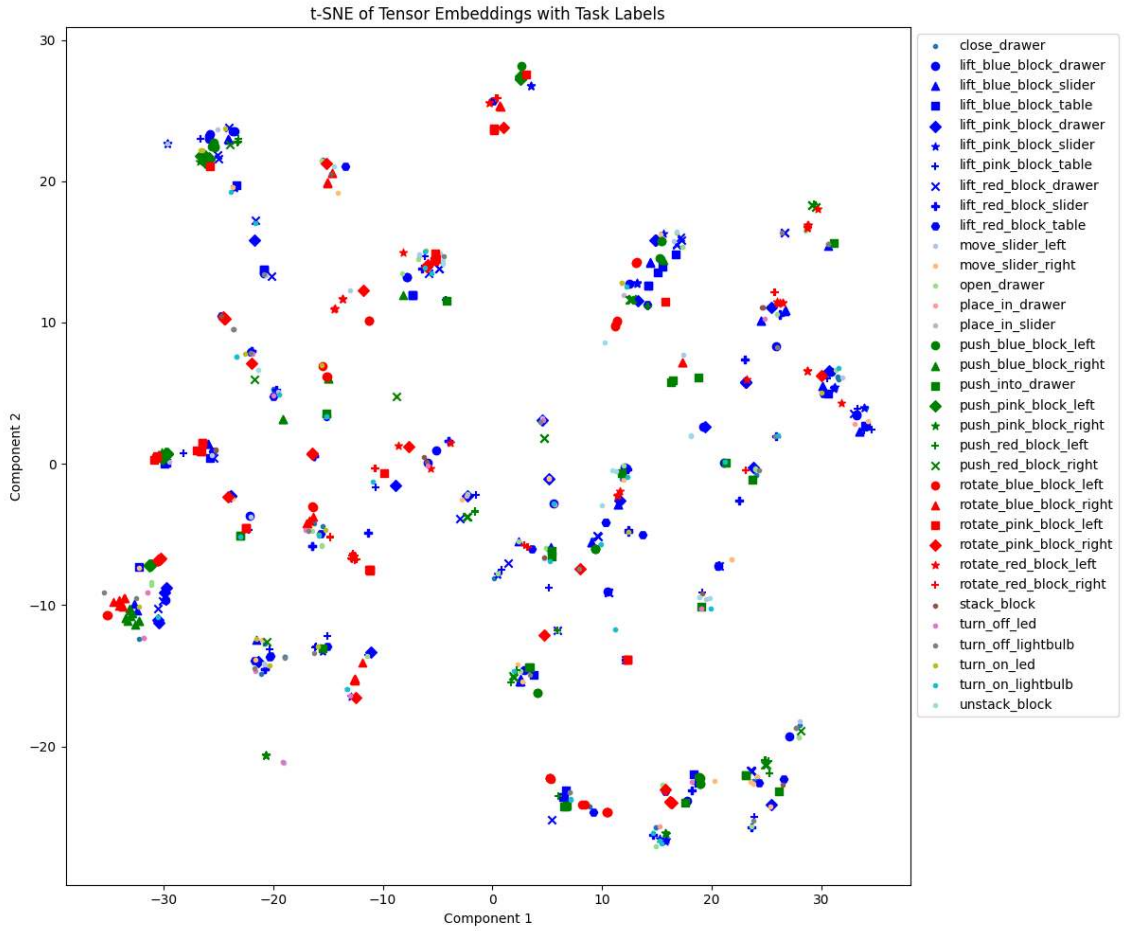

Figure 2.5: **t-SNE Visualization of State Embedder Embeddings.** t-SNE plots illustrate the final embeddings of the State Embedder, along with the corresponding clustering of these embeddings by action class.

Task	Definition
Rotate red/blue/pink block right/left.	The object has to be rotated clockwise/counter-clockwise more than 60° around the z-axis while not being rotated more than 30° around the x or y-axis.
Push red/blue/pink block right/left.	The object has to move more than 10 cm to the right/left while having surface contact in both frames.
Move slider left/right	The sliding door has to be pushed at least 12 cm to the left/right.
Open/close drawer	The drawer has to be pushed in/pulled out at least 10 cm.
Lift red/blue/pink block table	The object has to be grasped from the table surface and lifted at least 5 cm high. In the first frame the gripper may not touch the object.
Lift red/blue/pink block slider	The object has to be grasped from the sliding cabinet's surface and lifted at least 3 cm. In the first frame the gripper may not touch the object.
Lift red/blue/pink block drawer	The object has to be grasped from the drawer's surface and lifted at least 5 cm high. In the first frame the gripper may not touch the object.
Place in slider/drawer	The object has to be placed in the sliding cabinet/ drawer. It must be lifted by the gripper in the first frame.
Push into drawer	The object has to be pushed into the drawer. It has to touch the table surface in the first frame.
Stack/Unstack blocks	A block has to be placed/removed on top of another block. It may not be in contact with the gripper in the final/first frame.
Turn on/off lightbulb/LED	The switch has to be pushed up/down to turn on/off the yellow/green lightbulb/LED.

Figure 2.6: **Task Labels and Descriptions in CALVIN Benchmark.** 34 different task labels and corresponding descriptions in the CALVIN Benchmark.

Chapter 3

ROBOT INSTRUCTION FOLLOWING

3.1 Introduction

Language-guided robot control is a cutting-edge area in robotics, where natural language instructions are used to direct a robot's actions to perform complex human-driven tasks. In our research, we work on this challenge within the unique constraints of a "blind robot" setting to motivate our research question[1] yet in another downstream task, relying on proprioceptive information without any visual input. This makes the task much harder because the robot has to understand and follow commands, and then interact with its environment correctly, all without any visual feedback.

This part of our study builds upon the research discussed in Section 2 by extending it to address more complex robot manipulation tasks. Specifically, our goal is to enable an agent to successfully complete sequences of up to five language instructions, consecutively. This extension poses a significant challenge since it requires advanced integration of language and sensory-motor experiences.

To reiterate, our hypothesis is rooted in the concept that language is deeply intertwined with sensory-motor experiences during early developmental stages, and that intelligence becomes embodied through an incremental and developmental process, much like it does in babies [Smith and Gasser, 2005]. To validate this hypothesis, we again align with the principles of SSL by emphasizing the incremental and developmental nature of human nature in learning.

In our approach, we first pretrain a State Embedder based on the GPT [Radford et al., 2019b] architecture from scratch, as we did in Section2. The primary aim is to learn the hidden latent representations within an agent's internal motor space; from

[1]Can we learn good and accurate representations from an agent's internal motor space?

an agent's cartesian coordinates, arm-angle, and gripper information. This representation is then fused with language representations derived from an LLM based on MPT [Team, 2023], so that we can leverage the power of LLMs for complex reasoning and instruction processing. For the multimodal fusion of these different data modalities, we employ the Flamingo architecture. Additionally, we experiment with various Proprioceptive Encoders, such as fully connected networks (FCs), multilayer perceptrons (MLPs), and recurrent neural networks (RNNs), as previously explored in our Action Classification study (see Section 2.4.2). This methodology pushes the boundaries of robot autonomy and adaptability, requiring sophisticated integration of language understanding, action planning, and proprioceptive feedback processing.

3.2 Related Work

To establish a comprehensive understanding of the baselines utilized in this chapter of our study, we first provide an overview of the foundational approaches and methodologies. Each baseline integrates both visual and linguistic inputs to optimize motor control actions. The following sections detail the architecture, input-output mechanisms, and optimization criteria of these models, offering insights into their unique contributions and comparative performance in our experimental setup. Also, allows the reader to understand the evolution of the related work.

3.2.1 MCIL

Approach. An agent designed to resolve robotic manipulation tasks according to specific instructions must effectively integrate perceptual and linguistic inputs to generate a sequence of precise motor commands that interact with its environment. To address this, Multi-context Imitation Learning (MCIL) is employed [Lynch and Sermanet, 2020], to derive a general-purpose goal-reaching policy from unstructured play data. The strategy utilizes a 'relabeled' dataset, $D_{play} = \{(\tau, x_g)_i\}_{i=0}^{D_{play}}$, where each goal state x_g corresponds to a trajectory rollout $\tau = \{(x_0, a_0), ...\}$ that provides demonstrations for achieving the specified goal. The primary objective is to learn a

goal-reaching policy, $\pi_\theta(a_t|x_t, g)$, which determines the appropriate action a_t at each time step t in the trajectory τ, based on the current state x_t and the goal state g.

Inputs and Outputs. MCIL leverages a goal-reaching policy from unstructured and 1% language-labeled data by training the policy $\pi_\theta(a_t|x_t, g)$. However, since only 1% of the dataset includes language labels, the policy formulation primarily addressed is $\pi_\theta(a_t|x_t, x_g)$ where x_g represents the goal image when a language label is unavailable, and $\pi_\theta(a_t|x_t, l)$ when a language label l is provided. The model aims to generate actions that are either absolute Cartesian movements relative to the world frame or relative Cartesian displacements concerning the gripper frame. Consequently, the model inputs consist of a goal image (world frame), robot observations such as end-effector (EEF) positions and orientations, gripper width, joint positions, and gripper actions, and language labels.

Architecture. MCIL addresses the inherent multi-modality[2] present in free-form imitation datasets by encoding contextual demonstrations into a *latent plan* space using a sequence-to-sequence conditional variational auto-encoder (cVAE) [Sohn et al., 2015]. This approach forces the model to learn a plan z that teaches how to navigate from x_t to x_g through conditional distributions. This is crucial for addressing the multi-modality challenges in embodied tasks, where there is no singular path or complete rollout for a specific task; instead, the model learns the distribution of potential sub-actions. Subsequently, a decoder, functioning as a policy, is trained to reconstruct the input actions, conditioned on the current state x_t, goal state x_g, and the inferred plan z. MCIL consists of several components including (i) a Perceptual Encoder (for RGB images, Depth images, and robot state observations), (ii) a Language Goal Encoder, (iii) a Visual Goal Encoder, and an Action Decoder.

Criterion. Demonstrations conditioned on short-horizon goal images are utilized within a straightforward goal-conditioned imitation objective for maximum likeli-

[2]Here, multi-modality refers to the variety of possible actions, not the data type.

hood estimation:

$$\mathcal{L}_{LfP} = \mathbb{E}_{(\tau, x_g)\sim D_{\text{play}}} \left[\sum_{t=0}^{|\tau|} \log \pi_\theta(a_t \mid x_t, x_g) \right] \tag{3.1}$$

This approach facilitates the optimizes a goal-reaching policy, $\pi_\theta(a_t|x_t, g)$. During each training step, two types of contextual imitation losses are computed: (i) one for image goals and (ii) another for language goals. Each type involves minimizing the **Kullback-Leibler (KL)** divergence between the posterior and prior distributions and computing the **maximum likelihood for action reconstruction** loss by decoding plans into specific actions.

3.2.2 HULC

Approach. HULC [Mees et al., 2022a] is an enhanced version of MCIL that utilizes a static camera to generate global plans and learn local policies through a gripper camera conditioned on the plan. Similar to MCIL, the key insight is to solve a single imitation learning policy for either provided goal images or language goals, enabling control learning primarily from unlabeled play data.

Inputs and Outputs. HULC takes as input the gripper image, a third-person view image, and language, and produces discretized 7-DoF relative action outputs for the EEF.

Architecture. HULC comprises several components and features a somewhat complex architectural design. It includes separate vision encoders for the gripper and static cameras, a language encoder, a plan sampler network, a multimodal transformer, and a local policy network. Language instructions and visual observations are encoded first. During training, a multimodal transformer processes sequences of observations to learn and organize high-level behaviors through a posterior. These temporally contextualized features serve as input to a contrastive visuo-lingual alignment loss. The plan sampler network takes the initial state and latent language goal to predict the distribution of plans for achieving the goal. Both prior and posterior distributions are represented as vectors of multiple categorical variables and

are trained by minimizing KL divergence. The local policy network uses the latent language instruction, gripper camera observation, and global latent plan to generate a sequence of relative actions in the gripper camera frame to achieve the goal.

Criterion. It optimizes three different auxiliary losses: contrastive loss, KL loss, and action loss. The contrastive loss maximizes the cosine similarity between the visual features of sequence i and the corresponding language features while minimizing the cosine similarity between the current visual features and other language instructions in the same batch. This contrastive loss is defined similarly to the one used for pairing images and captions in CLIP. For the action loss, the action space is discretized and the policy is parameterized as a discretized logistic mixture distribution. Each of the predicted k logistic distributions has a separate mean and scale, and is weighted to form the mixture distribution. The imitation loss is the negative log-likelihood for this distribution.

3.2.3 RT-1

Approach. Robotics Transformer 1 (RT-1) [Brohan et al., 2022] is a highly efficient model designed for practical robotic control. It can process vast amounts of data, generalize effectively, and generate real-time actions. RT-1 uses a short sequence of images and natural language instructions as inputs, producing corresponding actions for the robot at each time step.

Inputs and Outputs. It takes six images as temporal input, along with the corresponding task description, and generates discrete actions.

Architecture. RT-1 integrates a Vision Encoder, a Text Encoder, FiLM layers, a Token-Learner, and a Transformer architecture. In more detail, the Vision Encoder processes a sequence of 6 images using an ImageNet pre-trained EfficientNet-B3 model. Each image has a resolution of 300 x 300, and the model outputs a spatial feature map of 9 x 9 x 512 from its final convolutional layer. This feature map is then flattened into 81 visual tokens for subsequent network layers. For Text Encoder the

instructions, a Universal Sentence Encoder is used. Film Layers are used to combine image and instruction tokenization through EfficientNet-B3, involving 26 layers of MBConv blocks and FiLM layers. This process generates 81 vision-language tokens and consists of a total of 16 million parameters. To enhance efficiency and speed up inference, RT-1 employs TokenLearner. This component compresses the 81 fused tokens into 8 final tokens, which are then passed to the Transformer layers. The 8 tokens per image are concatenated with tokens from other images in the sequence, resulting in 48 total tokens (with position encoding) fed into the Transformer backbone. The Transformer is a decoder-only sequence model with 8 self-attention layers and 19 For action tokenization, each action dimension in RT-1 is discretized into 256 bins. The target value for each variable is mapped to one of these bins, which are uniformly distributed within the variable's range.

Criterion. The use of a categorical cross-entropy objective and causal masking in a Transformer-based controller for the action space involves predicting a probability distribution over discrete actions, comparing it to the true actions using cross-entropy loss, and ensuring temporal causality with masking. This setup helps the model learn to predict the correct actions over a sequence of inputs effectively.

3.2.4 Flamingo

Approach. Flamingo [Alayrac et al., 2022] is a multimodal Vision Language Model (VLM) designed to integrate vision and text modalities. It capably processes sequences of arbitrarily interleaved visual and textual data, applicable to both images and videos. The model employs a Vision Encoder and a Perceiver Sampler to manage large, varying-size feature maps, converting them into a smaller number of visual tokens. For text generation, it utilizes a frozen Decoder-based LLM, which is conditioned on the visual representations outputted by the Perceiver Resampler. Optimization is achieved through the incorporation of Gated Cross-Attention dense blocks within the original frozen layers. It establishes state-of-the-art results for few-shot learning across a diverse set of multimodal language and image/video understanding tasks.

Inputs and Outputs. On the input side, Flamingo accepts either an image or video paired with corresponding text. Equipped with its Vision Encoder and Perceiver Sampler, the model efficiently processes both images and videos of arbitrary lengths. For each image/video and text pair, Flamingo generates relevant text.

Architecture. Within its architecture, Flamingo comprises three key components: (i) a Vision Encoder, (ii) a Perceiver Sampler, and (iii) a Text Encoder. Vision Encoder is a pre-trained and frozen Normalizer-Free ResNet (NFNet). For images, the output from the final stage—a 2D spatial grid of features—is flattened into a 1D sequence. For videos, frames are sampled at 1 FPS and encoded independently, resulting in a 3D spatio-temporal grid of features. Learned temporal embeddings are then added, and the features are flattened to 1D prior to being processed by the Perceiver Resampler. This module links the Vision Encoder with the frozen LLM. It accepts a variable number of image or video features from the vision encoder and consistently outputs a fixed set of visual tokens (64). This standardization significantly reduces the computational complexity involved in the vision-text cross-attention process. The Text Encoder is based on a Frozen LLM, enhanced by the integration of additional Gated Cross-Attention Dense Blocks, which are the only trainable components inside LLM. This decoder-based LLM is conditioned on the visual representations produced by the Perceiver Resampler. The queries coming from LLM are cross-attend to the visual output from the Perceiver Resampler in these Gated Cross-Attention Dense Blocks.

Criterion. Flamingo objective minimizes the Negative Log-Likelihood objective for each token on the text side by adjusting weighting parameters for each dataset.

3.2.5 RoboFlamingo

Approach. RoboFlamingo leverages the pretrained Vision Language Model, Flamingo [Alayrac et al., 2022], by integrating an explicit policy head. It is subtly fine-tuned through imitation learning exclusively on language-conditioned manipulation datasets, specifically CALVIN [Mees et al., 2022b]. The concept is straightforward:

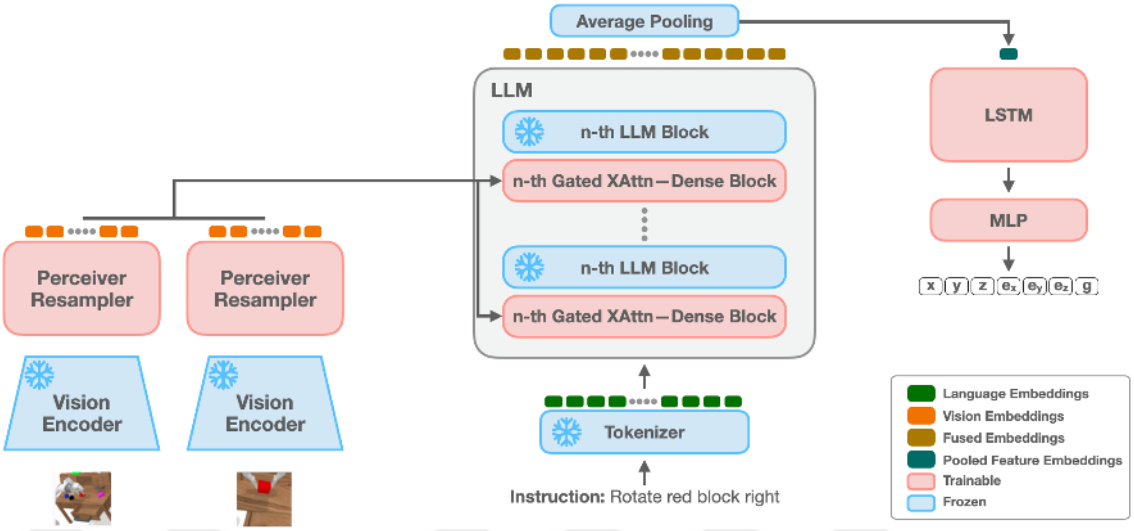

Figure 3.1: **RoboFlamingo Architecture.** The proposed RoboFlamingo module framework builds upon the Flamingo architecture, incorporating action decoding components such as Average Poolings, LSTM, and MLP. The red-colored blocks represent trainable modules, while the blue-colored blocks indicate frozen parameters. RoboFlamingo processes third-view and gripper-view images of the robot arm to generate direct actions, including transition, rotation, and gripper information. The framework utilizes ViT [Dosovitskiy et al., 2020] as the Vision Encoder and MPT [Team, 2023] as the LLM.

it utilizes existing VLMs with minimal fine-tuning on robotics data. Essentially, RoboFlamingo adds a policy head capable of utilizing historical data atop a robust VLM.

Inputs and Outputs. RoboFlamingo processes input consisting of third-person view images and gripper-view sequences, along with corresponding task descriptions in natural language. As output, it predicts the 7 DoF EEF pose and the gripper's status as open or closed.

Architecture. In its model settings, RoboFlamingo is an extension of the pre-trained Flamingo model, following the fine-tuning paradigm of OpenFlamingo [Awadalla et al., 2023]. It achieves this by training only the parameters of the resampler, the gated cross-attention module of each decoder layer, and the policy head while freezing all other parameters. For Vision Encodings, RoboFlamingo uses a Vision Transformer (ViT) as its vision feature extractor. It encodes the vision input sequences

into visual token sequences at each time step t. It is frozen during training. The Perceiver Resampler module acts as a token resampler, mapping the vision features to a fixed latent space to reduce the number of tokens before integrating them into the LLM as Key (K) and Value (V) pairs. It is trained and optimized during the training process. The Text Encoder is a decoder-based LLM, specifically MPT as used in OpenFlamingo. Gated cross-attention layers are inserted and updated during training, while the LLM parameters remain frozen. The cross-attention layer uses the language token as a Query and the encoded visual token as Key and Value, fine-tuned with imitation learning objectives on manipulation data. The multimodal fusion operation is performed within the gated cross-attention mechanisms. This deep interaction between vision and language tokens is expected to yield an informative vision-language joint embedding for robot manipulation. Finally, a Policy Head component converts the fused feature representations of vision observations and language instructions, which is the output of the LLM, into low-level control signals. It employs an LSTM with an MLP on top to map LSTM output embeddings to the 7 DoF action space. The fused features from the LLM undergo a Max Pooling operation and are then fed into the LSTM, followed by an MLP, which predicts the model's action outputs.

Criterion. The desired relative pose is optimized via regression loss with mean squared error (MSE) loss and the gripper status uses classification loss as binary cross-entropy (BCE) loss:

$$\mathcal{L} = \sum_t MSE(a_t^{pose}\hat{a}_t^{pose}) + \lambda_{gripper} BCE(a_t^{gripper}\hat{a}_t^{gripper}) \tag{3.2}$$

Notably, our work extends these baselines by building on top of RoboFlamingo and introducing a novel direction that leverages the Flamingo module. Unlike previous approaches that rely on visual data, our model utilizes robot proprioceptive data, marking an evolution in the related work to further enhance performance in language-conditioned manipulation tasks.

3.3 CALVIN Environment

As discussed in Section 2.2, we use the CALVIN dataset [Mees et al., 2022b] and environment, but in this part of our study, we focus on robot manipulation tasks within a robot simulation environment. Here, the agent relies on perceptual observations from its own proprioception states, making the task more challenging as it requires real-time decision-making rather than the supervised action classification described in Chapter 2. During our experiments, we again utilize CALVIN, for learning long-horizon language-conditioned tasks. This environment and its corresponding datasets serve as our imitation learning demonstration data. As stated, it includes 34 distinct tasks and evaluates 1000 unique instruction chains for sequential tasks. During each experiment, the robot must successfully complete sequences of up to five language instructions consecutively.

CALVIN combines the challenging aspects of open-ended robotic manipulation with open-ended human language conditioning. For instance, a robot instructed to "place the pink block inside the drawer" must relate the language to its world model, identify the blue block and drawer in its multimodal perceptual observations, and determine the best sequence of actions to complete the task. Ideally, a general-purpose robot should be capable of performing any combination of tasks instructed with natural language in any order.

This evaluation aims to verify how well the learned multi-task language-conditioned policy can accomplish several language instructions in a row. The 34 tasks from the previous evaluation are treated as subgoals, and valid sequences consisting of five sequential tasks are computed. Only feasible sequences that can be achieved from a predefined initial environment state are allowed. The evaluation sequences are filtered for cycles, redundancies, and similarities, resulting in 1000 unique instruction chains. For example, sequences like close the drawer"..."place in drawer" (unfeasible), "move slider right"..."move slider left"..."move slider right" (cyclic), or "push blue block left"..."push red block left" (similar) are excluded. The robot is reset to a neutral position after every sequence to avoid biasing the policies through the robot's initial pose. Different initial scene configurations are included in the eval-

uation to better assess generalization capabilities. For each subtask, the policy is conditioned on the current language instruction, and the transition to the next subgoal occurs only if the agent successfully completes the current task according to the environment's state indicator.

3.4 Model

As highlighted, our research examines a "blind robot" setting relying only on proprioception, without visual input. This increases task difficulty as the robot must interpret commands and generate actions using only its motor space. We combine LLMs for reasoning with a pre-trained State Embedder via Gated-XAttention adapters [Alayrac et al., 2022, Awadalla et al., 2023] to process robot state observations. Figure 3.2 illustrates our Blind Robot framework.

In our approach, we begin by pre-training a State Embedder from scratch, utilizing the GPT architecture, following Section 2 but with different inputs. The primary objective is to capture the hidden latent representations in an agent's motor motion space, which includes the Cartesian coordinates, arm angle, gripper status, the 7-DoF arm joints and sometimes the scecen states according to the experiment. These representations are then integrated with language representations derived from an LLM based on MPT [Team, 2023]. For the effective multimodal fusion of these disparate data types, we employ the Flamingo architecture [Alayrac et al., 2022, Awadalla et al., 2023]. Specifically, we incorporate Gated XAttn-Dense blocks between the existing LLM blocks [Alayrac et al., 2022]. These blocks are trained from scratch which lets the backpropagation occur solely within these dense blocks, thereby enhancing the integration of multimodal information.

In addition to the pre-trained State Embedder, we experiment with various Proprioceptive Encoders, such as fully connected networks, MLPs, and RNNs which are trained from scratch, as previously explored in our Action Classification study. Refer to Section 2.4.2 for details.

Model	Parameters	$d_{input/output}$	Layers	Heads	d_{model}	Context Size	# of samples
State Embedder RO	1.7M	15	8	8	128	1024	2.3M
State Embedder SO	1.7M	24	8	8	128	1024	2.3M
State Embedder RSO	1.7M	39	8	8	128	1024	2.3M

Table 3.1: **State Embedder Architecture.** Architecture and optimization hyperparameters for the proposed State Embedders that are pre-trained from scratch.

3.4.1 Pre-training State Embedder

Input Features. We pre-trained three different State Embedders based on their input feature types to determine which internal robot representations work best and could potentially replace vision as a form of perception. Firstly, we pre-trained autoregressively using only the Robot Observations (RO), which comprise a 15-dimensional vector. This vector includes the robot's tool center point position (3-dimensional) and orientation (3-dimensional) in world coordinates, gripper opening width in meters (1-dimensional), arm joint states (7-dimensional) in radians, and gripper action (1-dimensional) as a binary value indicating either open or closed. Secondly, we pre-trained using the Scene Obervations (SO), which includes the position and orientation of all objects in the scene. This results in a 24-dimensional vector comprising the joint states of the sliding door, drawer, button, and switch (4-dimensional), the on/off status of the lightbulb and green light (2-dimensional), and the position and orientation information of the red, blue, and pink blocks (18-dimensional). Finally, we combined both the robot and scene observations (RSO) by concatenating these two states to form a 39-dimensional state information vector.

Architecture. We implemented a State Embedder, similar to the one described in Section 2.3.2, following GPT architecture [Radford et al., 2019a, Radford et al., 2019b, Brown et al., 2020]. Unlike in traditional language modeling, which focuses on predicting the next language token, our model predicts action tokens represented by 15-dimensional vectors for RO inputs, 24-dimensional vectors for SO input, and 39-dimensional vectors for the concatenation of these states. The details of each State Embedder for varying input features can be seen in Table 3.1. The input and

output features consist of state observations, and the training data is sourced from State Trajectories with a window size of 1024 (See Figure A.2). The decoder has an embedding dimension of 128 and comprises 8 transformer layers, each with 8 attention heads. The model applies a dropout rate of 25% to mitigate overfitting and uses the GELU activation function [Hendrycks and Gimpel, 2016]. The feed-forward method within the transformer blocks is fully connected, and the model is trained using the Mean Squared Error (MSE) loss function which is appropriate for predicting continuous action values.

Pre-training. The pre-training of our GPT-based State Embedder is an autoregressive process in 1024 length of rollout windows, trying to generate the next state information of the agent. During training, we used a batch size of 16 to balance computational efficiency and model performance. The learning rate is initially set to 5e-4 and is adjusted using a learning rate scheduler which multiplies the learning rate by a factor of 0.8 if no improvement is observed over 3 consecutive epochs. Additionally, the learning rate scheduler has no cooldown period and allows the learning rate to decrease to a minimum of 1e-6 to ensure sustained learning throughout the training period. We employ the Adam optimizer with beta values set to 0.9 and 0.999 respectively, which leads to a stable and efficient convergence. A weight decay of 0.01 is applied to regularize the model and prevent overfitting. To further ensure stability, gradient clipping is implemented with a maximum value of 1.0 to prevent gradient explosion during backpropagation. The model is trained over 300 epochs to allow sufficient time for the model to learn and generalize from the data. Training is conducted with a precision of 32 bits. Gradient accumulation is set to 1, meaning the model updates weights after every batch. We The training was conducted on a single Nvidia A40 GPU and completed in approximately 3 hours. Furthermore, the entire training process is integrated with Weights & Biases (wandb) for comprehensive experiment tracking and visualization.

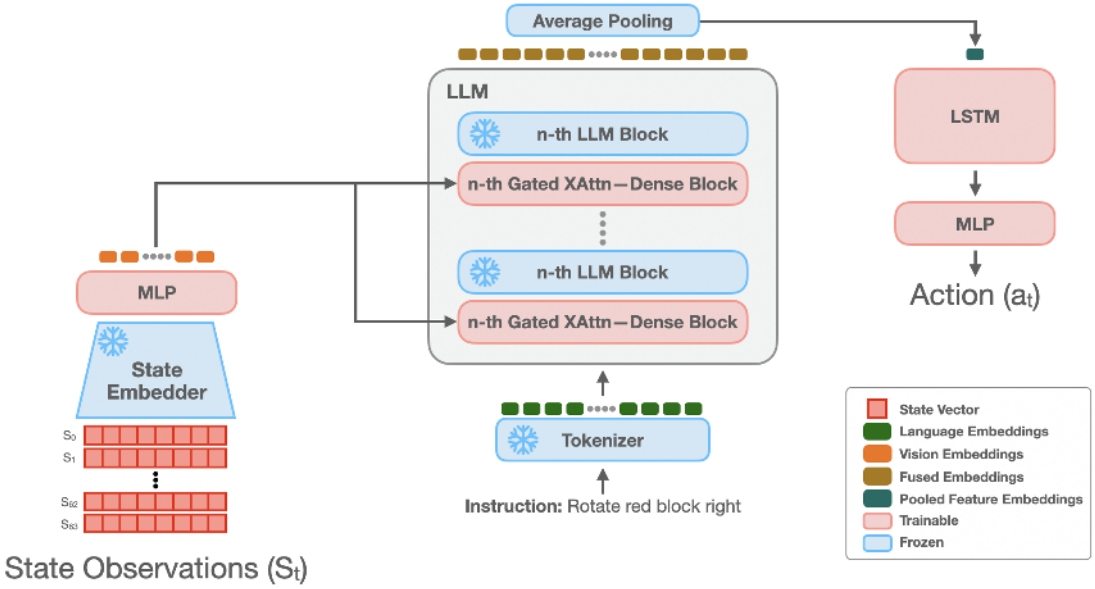

Figure 3.2: **Proposed Blind-RoboFlamingo Framework.** This figure illustrates the architecture designed to enable a blind robot to interpret and execute linguistic commands using only its internal motor space representations, without visual input. Proprioceptive data, including the robot's tool center point position, orientation, gripper opening width, arm joint states, and gripper action, are encoded with the pre-trained State Embedder. These encoded representations are then integrated with language instructions via Gated Cross-Attention (XAttn)-Dense blocks for fusion. The model predicts action tokens using an LSTM and an MLP on top, functioning as a Policy Head. Red-colored blocks represent trainable modules, while blue-colored blocks indicate frozen parameters. Blind-RoboFlamingo processes state information as input and generates direct actions, including transition, rotation, and gripper movements of the EEF. The framework leverages a pre-trained State Embedder described in Section 3.4 and utilizes MPT [Team, 2023] as the language model.

3.4.2 Finetuning Blind-RoboFlamingo

In that part of our research, as previously motivated, we aim to develop better abstract state representations for embodied action modeling, potentially replacing vision as a form of perception. To achieve this, we first pre-trained a State Embedder on a large unlabeled corpus within CALVIN environments (including all A, B, C, and D environments) as explained in Section 3.4.1. Now in this section, we integrate the pre-trained State Embedder into the RoboFlamingo framework, eliminating the need for a visual encoder and using the State Embedder representations instead,

as shown in Figure 3.2. The following sections detail the integration of this State Embedder, including the mathematical formulation and fine-tuning settings.

Blind-RoboFlamingo. We updated the RoboFlamingo backbone f_θ to process state and language inputs at each decision step, referring to this version as *Blind-RoboFlamingo*[3]. This model encodes state observations into latent tokens using the pre-trained State Embedder, then fuses these tokens with language targets through the feature fusion decoder, as illustrated in Figure 3.2.

State Embedder. As previously mentioned, the State Embedder comprises 8 decoder layers followed by a fully connected layer that maps the hidden dimensions to the token space. At each time step t, the provided states S_t are encoded into $\hat{X}_t$ using the pre-trained State Embedder:

$$\hat{X}_t = Decoder(S_t) \tag{3.3}$$

During fine-tuning, the State Embedder remains completely frozen. Then, another fully connected layer upscales the state space to align with the LLM's embedding space, and this layer is trainable during fine-tuning. Formally, this operation is defined as:

$$X_t = FC(\hat{X}_t) \tag{3.4}$$

where X_t is in the same latent space with the LLM.

Action with Fusion LLM. The action tokens produced by the fully connected layer are subsequently passed to the feature fusion decoder. This decoder is designed to create a state-language joint embedding by fusing the language instructions with the encoded state feature X_t. As in RoboFlamingo [Li et al., 2023b], we used the pre-trained decoder from OpenFlamingo [Awadalla et al., 2023] and fine-tuned the decoder module following the methods used in both OpenFlamingo and

[3]It is called "blind" because it does not rely on vision information, instead using its internal state information as perception to complete tasks

RoboFlamingo. Specifically, the decoder consists of L number of layers, each comprising a transformer decoder layer and a trainable cross-attention layer. The transformer layers are directly taken from a pre-trained language model, MPT [Team, 2023] in our case, and remain frozen throughout the training process. The cross-attention layer uses the language token as the query and the encoded state token as the key and value matrices, and it is fine-tuned using imitation learning objectives on the robot manipulation data from CALVIN.

More mathematically, let $x_i \in \mathbb{R}^d$ be the i-th embedded token of the language, M be the length of the language token, and $X \in \mathbb{R}^{M \times d}$ be the embedded matrix of the language tokens. The embedded instruction is then $X = (x_1, x_2, \cdots, x_M)$. The output X_t^{l+1} of the l-th decoder layer, given input X_t^l, is computed as follows:

$$\begin{aligned} \hat{X}_t^l &= \text{Tanh}(\alpha) \cdot \text{MLP}(A(X_t^l W_Q^C, X_t^l W_K^C, X_t^l W_V^C)) + X_t^l, \\ X_t^{l+1} &= \text{MLP}(A(\hat{X}_t^l W_Q^S, \hat{X}_t^l W_K^S, \hat{X}_t^l W_V^S)) + \hat{X}_t^l, \end{aligned} \tag{3.5}$$

where $X_t^1 = X$, $\hat{X}_t^l$ is the output of the gated cross-attention layer at time t, and $W_Q^C, W_K^C, W_V^C \in \mathbb{R}^{d \times d}$ are the learnable parameters of the cross-attention layer. The parameter $\alpha \in \mathbb{R}$ is a learnable gate parameter for stability control. The matrices $W_Q^S, W_K^S, W_V^S \in \mathbb{R}^{d \times d}$ are the parameters of the self-attention layer, and MLP denotes a multi-layer perceptron network. Through the interaction of state and language tokens, we expect the output $X_t = X_t^L = \{x_{t,1}^L, x_{t,2}^L, \cdots, x_{t,M}^L\}$ at each step t to be an accurate state-language embedding for the downstream task [Li et al., 2023b].

Action Policy Head. Following RoboFlamingo, we introduced an additional policy head p_θ to predict the action, such as the 7-DoF EEF transition, rotation, and gripper status. This is achieved by translating the output X_t^L embedding, which represents the state observation and language instruction. Using an LSTM with the state and language joint representation X_t^L, we obtain an aggregated embedding through a max-pooling operation over the token dimension and predict the action as follows:

$$\tilde{X}_t = \text{MaxPooling}(X_t) \tag{3.6}$$

$$h_t = \text{LSTM}(\tilde{X}_t, h_{t-1}) \tag{3.7}$$

$$a_t^{transition}, a_t^{rotation}, a_t^{gripper} = \text{MLP}(h_t) \tag{3.8}$$

where h_t is the hidden vector and $a_t^{transition}, a_t^{rotation}, a_t^{gripper}$ are the predicted end-effector transition, rotation, and gripper status at each time step t.

Criterion. We adopted the imitation loss objective from RoboFlamingo, optimizing the desired relative transition and rotation using MSE loss, and the gripper status using binary classification loss:

$$\ell = \sum_t \text{MSE}(a_t^{transition}, \hat{a}_t^{transition}) + \text{MSE}(a_t^{rotation}, \hat{a}_t^{rotation}) + \lambda_{gripper}\text{BCE}(a_t^{gripper}, \hat{a}_t^{gripper}) \tag{3.9}$$

where $\lambda_{gripper}$ is a hyperparameter that weights the gripper loss.

Training and Hardware Settings. We fine-tuned Blind-RoboFlamingo for one epoch using FP32 precision, with a batch size of 6 and no gradient accumulation. We employed the Adam optimizer with a constant learning rate of 0.0001 and 5000 warm-up steps. Due to the large-scale models and their substantial memory requirements, we conducted our training using eight NVIDIA A100 (80GB) GPUs in parallel, which proved challenging. The hardware specifications for each model and baseline used in the experiments, along with training durations are presented in Table 3.3.

3.5 Experiments and Results

We have experimented with several perspectives to examine the proposed Blind-RoboFlamingo method. First, we investigate how different input modalities, such as vision, language, and action states, impact task performance and explore their connection to the importance of abstract state representations. We evaluate the model's performance on language-conditioned robot manipulation tasks within the CALVIN environment [Mees et al., 2022b, Mees et al., 2022a].

Model	Trained Parameters	GPU Type	GPU Count	Training Hours
State Embedder RO	1.7M	A40 (80GB)	1	3
State Embedder SO	1.7M	A40 (80GB)	1	3
State Embedder RO + SO	1.7M	A40 (80GB)	1	3
RoboFlamingo	1.388B	A100 (80GB)	8	26
RoboFlamingo RO	1.396B	A100 (80GB)	8	28
RoboFlamingo SO	1.396B	A100 (80GB)	8	28
RoboFlamingo RO + SO	1.396B	A100 (80GB)	8	28
Blind-RoboFlamingo RO (FC)	0.892B	A100 (80GB)	8	10
Blind-RoboFlamingo RO (MLP)	0.892B	A100 (80GB)	8	10
Blind-RoboFlamingo RO (RNN)	0.892B	A100 (80GB)	8	10
Blind-RoboFlamingo RO (State Embedder)	1.033B	A100 (80GB)	8	14
Blind-RoboFlamingo RO + SO (State Embedder)	1.033B	A100 (80GB)	8	14

Table 3.2: **Device Overview of pre-training and Finetuning Model Configurations.** A detailed comparison of our model variants, highlighting the diversity in model sizes, the GPU hardware employed, the number of GPUs utilized, and the total hours of training required.

Model	Trained Parameters	GPU Type	GPU Count	Training Hours
State Embedder RO	1.7M	A40 (80GB)	1	3
State Embedder SO	1.7M	A40 (80GB)	1	3
State Embedder RSO	1.7M	A40 (80GB)	1	3

Table 3.3: **Device Overview of pre-training and Finetuning Model Configurations.** A detailed comparison of our model variants, highlighting the diversity in model sizes, the GPU hardware employed, the number of GPUs utilized, and the total hours of training required.

3.5.1 Benchmark

As previously discussed, we use the CALVIN benchmark as our testbed which is an open-source simulated environment designed for learning long-horizon language-conditioned tasks. It includes 34 distinct tasks and evaluates 1000 unique instruction chains for sequential tasks. In each experiment, the robot must successfully complete sequences of up to five language instructions consecutively, with the policy for each task dependent on a goal instruction. The agent advances to the subsequent goal only upon successful completion of the current task. The Long-Horizon Multi-Task

Language Conditioning (LH-MTLC) evaluation tests how well the learned policy can accomplish several language instructions in a row, using valid sequences of five sequential tasks. These sequences are filtered for feasibility, avoiding cycles, redundancies, and similarities, resulting in 1000 unique instruction chains. The robot is reset to a neutral position after each sequence to avoid biases, and different initial scene configurations are included to better evaluate generalization capabilities. The dataset is segmented into four divisions—labeled A, B, C, and D—with each containing 6 hours of human-teleoperated recording data, totaling over 2 million steps, of which only 1% (approximately 24,000 steps) are annotated with language instructions. Due to computational constraints, during our experiments, we utilized only the D environment for training our model and employed a separate validation split to test model performance.

3.5.2 Baseline

Our goal is to demonstrate whether we can develop better state representations for a robotic agent in embodied sequence modeling. To achieve this, we employed an SSL approach. First, we pre-trained a State Embedder on the unlabeled play split of the CALVIN dataset. Then, during the fine-tuning stage of Blind-RoboFlamingo, we replaced RoboFlamingo's Vision Encoder with our state-based State Embedder and fine-tuned it on labeled %1 data of CALVIN. To evaluate the effectiveness of our method, we implemented several baseline models and compared their performance on the CALVIN Benchmark:

1. RoboFlamingo: It is the original RoboFlamingo Framework illustrated in Figure 3.1.

2. RoboFlamingo (RO, SO, RSO): These baselines are extended versions of RoboFlamingo framework that incorporate state representations from Robot-Observations (RO), Scene-Observations (SO), and Robot-Scene-Observations (RSO). These state observation vectors are processed through a fully connected layer and merged with vision features produced by the Perceiver Sampler.

3. Blind-RoboFlamingo (RO, SO, RSO) with (FC, MLP, RNN): In these variants, the Vision Encoder is omitted (why it is called as "blind"), and state representations are encoded using a fully connected layer (FC), a three-layer multilayer perceptron (MLP) with ReLU nonlinearity, and vanilla recurrent neural networks (RNNs). These embeddings are then combined with language instructions inside LLM via Gated XAttn-Dense Blocks to fuse each state representation. We experimented with each state input feature as Robot-Observations (RO), Scene-Observations (SO), and Robot-Scene-Observations (RSO).

3.5.3 Results

In our experiments, we aim to disentangle the complexities and capabilities of robotic agents when enhanced with advanced state representation techniques. Our experiments are specifically designed to assess the impact of various modifications to the baseline model of RoboFlamingo in a controlled, simulated environment using the CALVIN dataset. All experiments were done using the CALVIN benchmark's D environment, emphasizing long-horizon, multi-task language-conditioned activities. Our investigation encompasses the following dimensions:

1. **The Importance of State Information:** We assess the sequential task completion capabilities of each RoboFlamingo model variant by training them on a consistent dataset. This enables us to determine the base effectiveness of each configuration in executing a series of robot manipulation tasks, as instructed through language.

2. **Blind-Robot Results:** We investigate the performance of RoboFlamingo configurations where vision encoders are disabled which we named as "Blind" RoboFlamingo. We evaluate how well these models perform in comparison to their sighted counterparts and highlight the role of non-visual state information in embodied tasks.

3. **Effect of the State Encoder Selection:** We compare the efficacy of vari-

ous state encoders, ranging from simple neural networks to sophisticated pre-trained models leveraged by SSL. We analyze the impact of these representation techniques on the performance of Blind RoboFlamingo configurations.

Through these perspectives, we aim to demonstrate not only the performance of these models but also their potential with sophisticated state encoding strategies to scale to more complex, real-world applications in robotic systems.

Method	Training Data	Test Split	Task Completed in a Sequence (Success Rate) 1	2	3	4	5	Avg Len
RoboFlamingo	D	D	0.692	0.235	0.058	0.012	0.001	0.998
RoboFlamingo RO	D	D	0.623	0.342	0.165	0.087	0.037	1.254
RoboFlamingo SO	D	D	0.643	0.315	0.114	0.042	0.015	1.129
RoboFlamingo RSO	D	D	0.677	0.415	0.234	0.138	0.077	1.541

Table 3.4: **The Importance of State Information in Task Sequences.** This table presents the success rates of different versions of the RoboFlamingo model across a sequence of five tasks. Each row represents a different model configuration, focusing on different combinations of state representations and architecture adjustments. Models are evaluated based on their success rates for each individual task (columns 1-5) and the average sequence length (Avg Len) they manage to achieve. The baseline configurations (RO, SO, RSO) integrate observations from Robot-Observations, Scene-Observations, and combined Robot-Scene-Observations. The effectiveness of each configuration is tested against 1000 unique instruction chains over 34 distinct tasks, with a focus on the robot's ability to adapt to different initial scene settings and to reset after each sequence to prevent biases.

The Importance of State Information. According to Table 3.4, the introduction of different state observations significantly affects the performance of the models. Comparing the baseline RoboFlamingo model, where the success rates decrease sharply from the first to the fifth task with an average sequence length of 0.998, we observe an improved task completion as more complex observation configurations are introduced. RoboFlamingo RO shows a notable improvement in later tasks, achieving an average sequence length of 1.254, indicating better persistence in task completion over sequences. RoboFlamingo SO displays slightly lower performance than RO, with an average sequence length of 1.129, suggesting that scene

observations alone are less effective than robot observations. RoboFlamingo RSO demonstrates the highest success rates across all tasks, with an average sequence length of 1.541. This model, combining both Robot and Scene observations, indicates that integrating multiple types of state information can substantially enhance the model's ability to handle complex sequences and let the model reason better using these two pieces of information together.

Method	Training Data	Test Split	Task Completed in a Sequence (Success Rate) 1	2	3	4	5	Avg Len
RoboFlamingo	D	D	0.692	0.235	0.058	0.012	0.001	0.998
Blind-RoboFlamingo RO with State Embedder	D	D	0.614	0.265	0.083	0.021	0.010	0.993
Blind-RoboFlamingo SO with State Embedder	D	D	0.620	0.218	0.046	0.014	0.001	0.899
Blind-RoboFlamingo RSO with State Embedder	D	D	0.657	0.302	0.137	0.053	0.028	1.177

Table 3.5: **Impact of State Embedder on Task Sequences with Vision Disabled.** This table compares the success rates of various Blind-RoboFlamingo models with original RoboFlamingo, across a sequence of five tasks. With vision encoders disabled, the "Blind" RoboFlamingo models generally show reduced performance compared to their sighted counterparts.

Blind-Robot Results. When vision encoders are disabled, resulting in "Blind" RoboFlamingo configurations, we see a general decrease in performance compared to their sighted counterparts. Blind-RoboFlamingo with State Embedder models (RO, SO, RSO) show reduced success rates compared to their non-blind versions but still perform respectably with an average sequence length of 0.993 with RO and 1.177 with RSO. Direct comparison with the baseline RoboFlamingo indicates that even blind models can outperform the baseline in certain configurations, emphasizing the strength of integrating diverse state information even without visual inputs. See Table 3.5

Effect of the State Encoder Selection. The performance difference between models using a State Embedder and those employing basic from-scratch training methods like FC, MLP, and RNN is substantial, check Table 3.6. Blind-RoboFlamingo models with simpler neural networks like FC and MLP struggle, with the MLP configuration barely achieving an average sequence length of 0.340. In contrast, our

Method	Training Data	Test Split	Task Completed in a Sequence (Success Rate) 1	2	3	4	5	Avg Len
Blind-RoboFlamingo with State Embedder	D	D	0.614	0.265	0.083	0.021	0.010	0.993
Blind-RoboFlamingo with FC	D	D	0.264	0.064	0.000	0.000	0.000	0.334
Blind-RoboFlamingo with MLP	D	D	0.267	0.062	0.010	0.001	0.000	0.340
Blind-RoboFlamingo with RNN	D	D	0.432	0.130	0.023	0.002	0.000	0.587

Table 3.6: **Effect of State Encoder Selection on Task Sequences.** This table compares the success rates of Blind-RoboFlamingo models with different state encoder configurations, including a State Embedder and basic neural network architectures (FC, MLP, RNN).

main model that leverages a pre-trained State Embedder demonstrates significantly better performance compared to other from-scratch methods, which underscores the importance of better state encodings. The use of a pre-trained State Embedder in the Blind-RoboFlamingo RO with an MLP on top of it results in a higher average sequence length of 1.057 compared to its simpler counterparts. That shows that well-trained state representations can extract and utilize critical information even from non-visual data. This highlights the necessity of developing robust state encoders that can generate informative representations from diverse inputs.

As a recap, our observations motivate the significant impact of diverse and different state information and sophisticated encoding techniques on model performance. The RoboFlamingo RSO model, incorporating both Robot and Scene observations, demonstrated better performance across all tasks which highlights the effect of integrating multimodal data types. In "blind" configurations, models utilizing our State Embedder outperformed simpler architectures and gave on-par results with the vision-based model. The effectiveness of our State Embedder is evident in its ability to extract meaningful representations from the robot's internal states using SSL, without any visual data. By grounding actions with language through our pre-trained State Embedder, we enabled further exploration into advanced state encoding techniques and the integration of recent advanced techniques to achieve better-embodied state representations that are used in robotic control systems.

3.6 Conclusion

Our work in language-guided robot control within a "blind robot" setting, which relies solely on proprioceptive information, has shown promising results in connecting language with action motor space. By extending foundational work to handle complex tasks involving up to five consecutive language instructions, we demonstrated that robots can effectively integrate language and sensory-motor experiences to perform intricate actions without visual input.

We pre-trained an State Embedder based on the GPT architecture to capture latent motor representations and fused these with language representations from an LLM using the Flamingo architecture. This multimodal fusion enabled sophisticated reasoning and action planning based on natural language commands. Experiments with various Proprioceptive Encoders, such as fully connected networks, MLPs, and RNNs, highlighted the impact of diverse state information on model performance. Our RoboFlamingo RSO model, integrating both robot and scene observations, outperformed simpler models and achieved results comparable to vision-based systems and our Blind-RobotFlamingo model leveraged by pre-trained State Embedder gives close results with Vision-based models.

The success of our State Embedder in extracting meaningful representations from internal states using SSL, without visual data, underscores the potential of grounding actions with language. This research advances the field by demonstrating the feasibility of using proprioceptive information for autonomous robotic control and opens new directions for future exploration in multimodal integration of embodied state representations.

Chapter 4

CONCLUSION

4.1 Conclusion

In this thesis, we have explored the interconnections between language and motor actions in robotics by particularly focusing on Robot Action Classification and Language-Guided Control within a proprioception-based setting.

In Chapter 2, our investigations into robot action classification have underscored its significance as a foundational aspect of AI. Unlike traditional methods that depend heavily on visual inputs, our approach has harnessed proprioceptive information to enable a "blind robot" to identify and categorize its actions. This shift towards an internal sensory-based understanding mimics the inherent sensory-motor coordination found in human cognitive development, aligning with the developmental trajectory of learning through incremental stages. By utilizing advanced Deep Learning strategies, such as transformers and SSL, we have significantly advanced the robot's ability to classify its actions with high accuracy in a constrained data environment. The implementation of these methods not only pushes the boundaries of robot perception but also enhances the robot's self-awareness and its interaction capabilities with the environment.

In Chapter 3, we dive into the second focal area of this thesis, language-guided robot control, which demonstrates how natural language can effectively steer robot actions in complex task settings without reliance on visual input as pixels during perception. By integrating language with the robot's motor representations through the use of LLMs and the Flamingo architecture, we have crafted a system where high-level language instructions are decomposed into actionable robotic tasks. This novel integration allows the robot to perform sequences of actions based purely on linguistic inputs and proprioceptive feedback, showcasing a significant leap in robotic

autonomy and adaptability. The successful implementation of this system highlights the potential for more intuitive representation in real-time decision-making.

Overall, this thesis not only shows the critical role of sensory-motor experiences in the development of cognitive capabilities but also showcases a new direction in robotic control through the lens of GLL. The achievements noted in robot action classification and language-guided control illustrate the transformative possibilities of integrating proprioceptive information with linguistic elements using SSL and recent techniques in LLMs by setting a robust foundation for future explorations in robotics.

4.2 Limitations and Future Work

While this thesis has made significant strides in advancing the integration of language and motor actions in robotics, several areas present opportunities for further research and development.

Enhanced State Encoders. To improve generalizability, future research could focus on the development of more advanced state encoders. This includes expanding the range of datasets beyond those used in Calvin, increasing model parameter sizes to capture more nuanced features, and exploring alternative architectural approaches. For instance, incorporating BERT-based models alongside existing decoders could enhance the encoding of state information and improve overall performance.

Utilization of Scene Information. The current approach provides complete scene information, which may lead to overfitting. Future studies could explore methods where only initial object locations are provided, allowing the robot to progressively memorize and adapt to changes in the environment. This could better simulate real-world scenarios and improve the robot's ability to handle dynamic settings.

Diverse Environments and Tasks. The research has primarily utilized static objects in the environment, which has led to overfitting and higher accuracy for these objects compared to dynamic ones. To address this, future work should incorporate a broader range of environments and tasks, including those with varying object dynamics. Testing across different environments, such as those found in the Calvin dataset (Environments A, B, and C) which will help assess the robustness of the models and their adaptability to new conditions.

Fusion Strategies and Advanced LLMs. The thesis utilized MPT-1B for LLM integration and the Flamingo architecture for multimodal fusion. Future research could experiment with more advanced LLMs, such as Mistral Large [Jiang et al., 2023], and explore task-specific instruction fine-tuning to enhance reasoning capabilities. Additionally, investigating newer multimodal architectures, such as LLaVA [Liu et al., 2024] or LLaMA-adapter [Gao et al., 2023], could provide improved fusion strategies and better performance in language-guided control.

This thesis explores a solid foundation for integrating language with robotic motor space and addressing these future work areas will further enhance the robustness, adaptability, and effectiveness of robotic systems in complex scenarios.

BIBLIOGRAPHY

[Achiam et al., 2023] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

[Agarap, 2019] Agarap, A. F. (2019). Deep learning using rectified linear units (relu).

[Ahn et al., 2022] Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., Yan, M., and Zeng, A. (2022). Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*.

[Alayrac et al., 2022] Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. (2022). Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.

[Awadalla et al., 2023] Awadalla, A., Gao, I., Gardner, J., Hessel, J., Hanafy, Y., Zhu, W., Marathe, K., Bitton, Y., Gadre, S., Sagawa, S., et al. (2023). Openflamingo: An open-source framework for training large autoregressive vision-language models. *arXiv preprint arXiv:2308.01390*.

[Belinkov, 2021] Belinkov, Y. (2021). Probing classifiers: Promises, shortcomings, and advances.

[Bengio et al., 1994] Bengio, Y., Simard, P. Y., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5 2:157–66.

[Bishop, 1995] Bishop, C. (1995). *Neural networks for pattern recognition*. Oxford University Press, USA.

[Brohan et al., 2022] Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., et al. (2022). Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*.

[Brown et al., 2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

[Cho et al., 2014] Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In Wu, D., Carpuat, M., Carreras, X., and Vecchi, E. M., editors, *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

[Chowdhery et al., 2023] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Dosovitskiy et al., 2020] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cogn. Sci.*, 14:179–211.

[Gao et al., 2023] Gao, P., Han, J., Zhang, R., Lin, Z., Geng, S., Zhou, A., Zhang, W., Lu, P., He, C., Yue, X., Li, H., and Qiao, Y. (2023). Llama-adapter v2: Parameter-efficient visual instruction model.

[Graves, 2012] Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer.

[Harnad, 1990] Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1–3):335–346.

[Hendrycks and Gimpel, 2016] Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[Hollich et al., 2000] Hollich, G. J., Hirsh-Pasek, K., Golinkoff, R. M., Rebecca, J., Brand, Brown, E., Chung, H. L., Hennon, E. A., Rocroi, C., and Bloom, L. (2000). Breaking the language barrier: an emergentist coalition model for the origins of word learning. *Monographs of the Society for Research in Child Development*, 65 3:i–vi, 1–123.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366.

[Jiang et al., 2023] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. (2023). Mistral 7b. *arXiv preprint arXiv:2310.06825*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA. Curran Associates Inc.

[Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324.

[Li et al., 2023a] Li, J., Li, D., Savarese, S., and Hoi, S. (2023a). Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR.

[Li et al., 2023b] Li, X., Liu, M., Zhang, H., Yu, C., Xu, J., Wu, H., Cheang, C., Jing, Y., Zhang, W., Liu, H., Li, H., and Kong, T. (2023b). Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*.

[Liu et al., 2024] Liu, H., Li, C., Wu, Q., and Lee, Y. J. (2024). Visual instruction tuning. *Advances in neural information processing systems*, 36.

[Lynch and Sermanet, 2020] Lynch, C. and Sermanet, P. (2020). Language conditioned imitation learning over unstructured data. *Robotics: Science and Systems XVII*.

[Mees et al., 2022a] Mees, O., Hermann, L., and Burgard, W. (2022a). What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters (RA-L)*, 7(4):11205–11212.

[Mees et al., 2022b] Mees, O., Hermann, L., Rosete-Beas, E., and Burgard, W. (2022b). Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334.

[Pascanu et al., 2013] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA. PMLR.

[Perez et al., 2018] Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. (2018). Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

[Radford et al., 2019a] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019a). Language models are unsupervised multitask learners.

[Radford et al., 2019b] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019b). Language models are unsupervised multitask learners.

[Raffel et al., 2019] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

[Rosenblatt, 1958a] Rosenblatt, F. (1958a). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

[Rosenblatt, 1958b] Rosenblatt, F. (1958b). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[Saffran et al., 1999] Saffran, J. R., Johnson, E. K., Aslin, R. N., and Newport, E. L. (1999). Statistical learning of tone sequences by human infants and adults. *Cognition*, 70(1):27–52.

[Samuelson et al., 2011] Samuelson, L., Smith, L., Perry, L., and Spencer, J. (2011). Grounding word learning in space. *PLoS One*, 6(12).

[Sarzynska-Wawer et al., 2021] Sarzynska-Wawer, J., Wawer, A., Pawlak, A., Szymanowska, J., Stefaniak, I., Jarkiewicz, M., and Okruszek, L. (2021). Detecting formal thought disorder by deep contextualized word representations. *Psychiatry Research*, 304:114135.

[Smith and Gasser, 2005] Smith, L. and Gasser, M. (2005). The development of embodied cognition: Six lessons from babies. *Artificial life*, 11:13–29.

[Sohn et al., 2015] Sohn, K., Lee, H., and Yan, X. (2015). Learning structured output representation using deep conditional generative models. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

[Sutskever et al., 2014a] Sutskever, I., Vinyals, O., and Le, Q. V. (2014a). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International*

Conference on Neural Information Processing Systems - Volume 2, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

[Sutskever et al., 2014b] Sutskever, I., Vinyals, O., and Le, Q. V. (2014b). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

[Team et al., 2023] Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. (2023). Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

[Team, 2023] Team, M. N. (2023). Introducing mpt-7b: A new standard for open-source, commercially usable llms. Accessed: 2023-05-05.

[Touvron et al., 2023] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

[van der Maaten and Hinton, 2008] van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

[Vaswani et al., 2023] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention is all you need.

[Vinyals et al., 2015] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Appendix A

A.1 State Trajectories

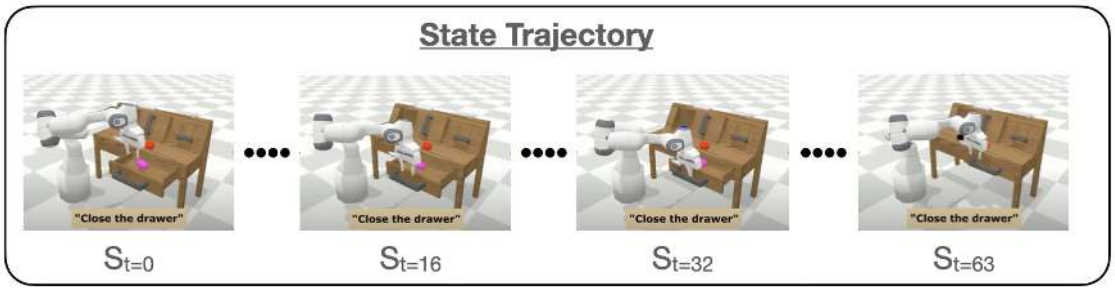

Figure A.1: **Labeld State trajectories for Downstream Tasks.** State trajectory with a window size of 64 from labeled data, illustrating the robot's execution of the instruction "Close the drawer," with state snapshots captured at key time steps during the rollout.

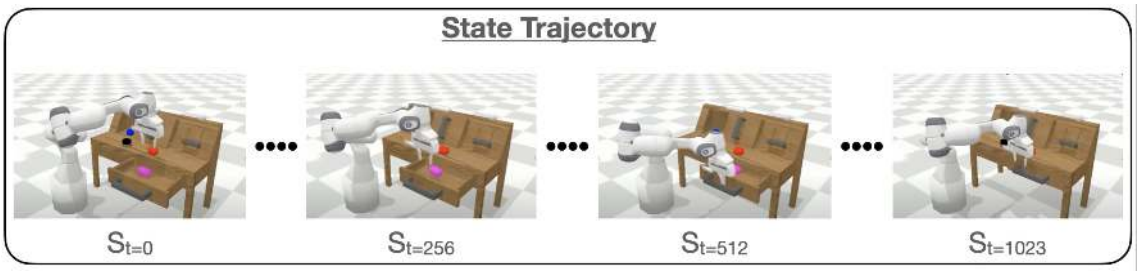

Figure A.2: **Unsupervised State trajectories for State Embedder Pre-training.** Unlabeled State trajectory with a window size of 1024, illustrating the robot's execution of random action with state snapshots captured at key time steps during the rollout.

A.2 State Observations: Robot Observations (RO) and Scene Observations (SO)

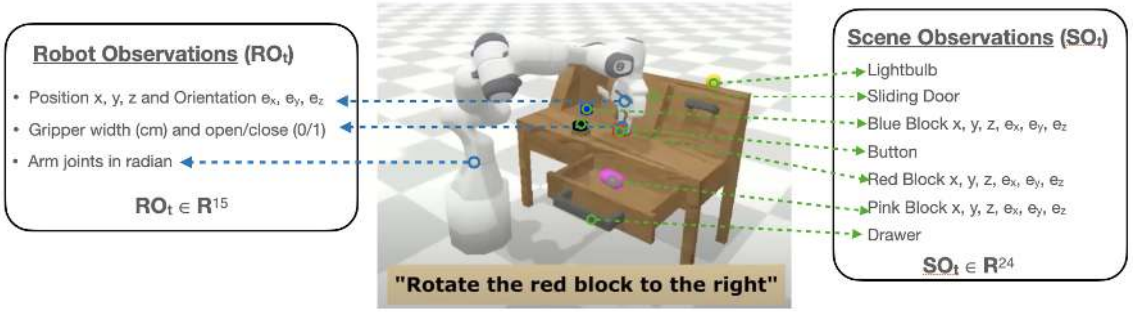

Figure A.3: **Detailed depiction of Robot and Scene Observations during the execution of the instruction "Rotate the red block to the right.".** The robot's observations (RO_t) include its position, orientation, gripper status, and joint angles, forming a 15-dimensional vector. Scene observations (SO_t) consist of positional and orientational data for various objects in the environment, such as the lightbulb, sliding door, and blocks, forming a 24-dimensional vector. These observations provide the necessary state information for the robot to interpret and act upon the instruction within the environment..

A.3 Affect of Input Features on Robot Action Classification

Method	n=1	n=2	n=4	n=all
SSL+Probing with Robot Observations	30.72	30.2	40.5	79.1
SSL+Probing with Scene Observations	55.1	62.7	66.7	86.1
SSL+Probing Scene-Differences (Ours)	**59.9**	**65.9**	**77.8**	**98.1**

Table A.1: **Affect of Input Features on Robot Action Classification.** Classification accuracies for CALVIN actions with varying input feature types.

Task	RoboFlamingo	RoboFlamingo RO	RoboFlamingo SO	RoboFlamingo RSO
lift_pink_block_slider	40.0%	48.2%	34.4%	48.5%
open_drawer	51.9%	85.3%	93.9%	87.7%
push_blue_block_left	40.0%	33.3%	41.2%	25.9%
push_red_block_left	37.3%	22.4%	41.0%	29.2%
place_in_slider	28.6%	82.6%	30.0%	80.0%
move_slider_left	77.1%	47.3%	48.1%	91.5%
turn_on_led	80.7%	81.6%	85.6%	98.1%
close_drawer	36.5%	87.2%	94.9%	94.1%
push_red_block_right	37.7%	43.1%	44.4%	37.1%
lift_red_block_slider	41.9%	48.3%	52.5%	45.6%
turn_on_lightbulb	63.7%	83.7%	86.6%	91.7%
lift_red_block_table	42.6%	67.6%	39.5%	65.1%
move_slider_right	82.7%	93.8%	78.0%	94.2%
lift_blue_block_table	59.3%	61.3%	33.3%	59.0%
lift_blue_block_slider	30.5%	49.2%	34.4%	60.0%
turn_off_lightbulb	68.8%	82.4%	73.1%	81.0%
place_in_drawer	48.7%	98.0%	28.6%	81.7%
lift_pink_block_table	60.7%	71.0%	42.1%	59.5%
rotate_pink_block_left	64.9%	7.7%	38.5%	38.6%
turn_off_led	52.4%	34.4%	67.4%	97.8%
rotate_pink_block_right	45.5%	19.0%	9.4%	6.2%
rotate_blue_block_left	26.2%	23.4%	47.6%	27.1%
rotate_red_block_right	36.7%	25.0%	16.3%	12.9%
push_pink_block_right	31.1%	58.3%	46.8%	57.9%
push_pink_block_left	57.4%	41.7%	76.8%	43.1%
rotate_red_block_left	21.6%	6.2%	40.5%	16.0%
push_into_drawer	37.2%	10.4%	16.7%	28.8%
stack_block	10.4%	10.2%	0.0%	4.0%
push_blue_block_right	26.7%	19.3%	26.1%	30.0%
unstack_block	100.0%	50.0%	100.0%	100.0%
rotate_blue_block_right	30.0%	24.1%	15.4%	10.8%
lift_blue_block_drawer	0.0%	100.0%	100.0%	60.0%
lift_red_block_drawer	0.0%	40.0%	0.0%	80.0%

Table A.2: **Instruction Following Accuracy by Task.** Success Rates (SR) for various tasks across different RoboFlamingo models with varying input features: RO (Robot Observations), SO (Scene Observations), and RSO (combined Robot and Scene Observations).