
School of Mechanical and Manufacturing Engineering
Faculty of Engineering
UNSW Sydney

BY

**Lucas Chu Barbosa**

# Real-World Residual RL for Vision-Language Robotic Manipulation

Thesis submitted as a requirement for the degree of Bachelor of Robotics and Mechatronics Engineering

| | |
|---|---|
| Submitted | 28/11/2025 |
| Student zID | z5259433 |
| Supervisor | Will Midgley |

**ORIGINALITY STATEMENT**

*'I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.'*

Signed: 

Date: 28/11/2025

Lucas Chu Barbosa: *Real-World Residual RL for Vision-Language Robotic Manipulation,* Deep-learning for Robotics, Bachelor of Engineering Honours and Computer Science ©

SUPERVISOR:
Will Midgley

LOCATION:
Sydney

## ABSTRACT

This thesis investigates the effectiveness of Reinforcement Learning (RL) algorithms used as residual adapters on top of large pre trained Vision-Language-Action (VLA) policies for real world robotic manipulation. A frozen VLA backbone provides a generalist visuomotor prior, while a compact RL controller learns an additive residual correction to the base action. The objective is not to show that RL outperforms the underlying VLA policy, but to determine which RL algorithms most effectively exploit a fixed pre trained policy in this residual setting. The study is conducted entirely on a physical SO-101 3D printed robotic arm controlled via teleoperation, without any use of simulation. Demonstrations collected through teleoperation are combined with on robot rollouts to train and evaluate multiple continuous control actor critic algorithms as residual adapters. A suite of tabletop reaching and manipulation tasks is used to compare these algorithms under variations in objects, viewpoints, and dynamics. Performance is assessed in terms of task success rate, adaptation speed, sample efficiency, and the stability and safety of online learning on hardware. The work provides a reproducible real robot pipeline and an evaluation protocol for benchmarking residual RL adapters on top of pre trained VLA policies, together with practical guidance for selecting RL backends for foundation model based robotic systems.

*If you want to go fast, go alone.*
*If you want to go far, go together.*

— African proverb

## ACKNOWLEDGMENTS

I am in full gratitude for the contributions, thoughtful discussions, and mentorship of all those who helped me on this journey. First, I would like to thank Will for his supervision, constructive guidance, and pragmatic feedback that consistently kept the project focused, grounded, and achievable. Your ability to balance ambition with realism helped steer this work in a direction that was both technically interesting and actually feasible.

Harry, thank you for the regular discussions on contemporary robotics research that sharpened my thinking throughout the project. Our conversations directly informed the literature review, the formulation of the research questions, and many of the design decisions in the experimental setup.

Jack, thank you for being a reliable sounding board throughout the early development of this project. Your openness to half formed ideas and your steady support through the long nights and early mornings proved invaluable.

Lachlan, I am especially grateful for your generosity in lending your GPU when my own hardware could not handle the demands of the thesis experiments. Without your willingness to share high quality inference hardware, this work would have been significantly delayed and far more stressful to complete.

To Jamie and my oNex team, thank you for supporting me throughout this journey. Building a company while completing a high quality thesis is not an easy combination, and your belief made it feel possible rather than unrealistic. Your patience with my shifting availability and your willingness to carry extra weight at key moments gave me the space to be both a committed founder and a serious researcher.

During my honors thesis I was also working full time at PwC as a Software Engineer and Research Lead, and at times the combination of a demanding role and serious research felt unmanageable. Tom and Rob made it possible to carry both. By supporting a shift to part time during the hardest stages of the thesis, they gave me the time and headspace I needed rather than forcing a choice between work and study. They did more than accommodate my situation; they supported me, trusted my judgment, and treated this thesis as work worth investing in. Their mentorship, understanding, and flexibility have shaped how I think about responsibility and leadership, and they directly influenced the quality of the projects I am able to execute today. For that, I am deeply and forever grateful.

Eric and Sam, thank you for your steady encouragement and for giving me the confidence that I could be a capable researcher. Your honest feedback always pushed me in the right direction rather than letting me settle for "good enough". You both inspired me to take research seriously and to consider a graduate path in this space. I have deeply valued our conversations on AI, the future, and technology, and those discussions have shaped both my worldview and the high level direction of this work.

I am also very thankful to my friends and family for their patience, encouragement, and understanding through long nights in the lab and many thesis driven schedule changes. Their support in the background made it possible for me to focus on the work in front of me. I owe a particular debt to my partner, Gabi, who has been there through thick and thin. From countless late study nights to a trip to Rotterdam while I tried to finalise the RL theory, she kept me steady, cared for, and reminded me that there is life outside the next experiment. Her patience and willingness to rearrange her own plans so that I could finish this work is something I do not take for granted. I am especially grateful to my parents. In particular, I thank my mum for caring so deeply about my progress, for worrying on my behalf when deadlines were close, and for always encouraging me to work harder. Your energy and belief in me have been a strong motivating force. I hope this work reflects, in part, the values you both instilled in me and marks a step toward learning to finish things a little earlier than the last minute.

To all of you, your support not only has materially improved the quality and clarity of this thesis, it has made the journey itself much more meaningful.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LISTINGS

# ACRONYMS

| | |
|---|---|
| A2C | Advantage Actor-Critic |
| A3C | Asynchronous Advantage Actor-Critic |
| AE | Autoencoder |
| AI | Artificial Intelligence |
| BERT | Bidirectional Encoder Representations from Transformers |
| BPE | Byte-Pair Encoding |
| CLIP | Contrastive Language-Image Pre-training |
| CNN | Convolutional Neural Network |
| DDPG | Deep Deterministic Policy Gradient |
| DoF | Degrees of Freedom |
| FiLM | Feature-wise Linear Modulation |
| GPT | Generative Pre-trained Transformer |
| GPU | Graphics Processing Unit |
| KL | Kullback-Leibler |
| LLM | Large Language Model |
| LSTM | Long Short-Term Memory |
| MAML | Model-Agnostic Meta-Learning |
| MDP | Markov Decision Process |
| MLM | Masked Language Model |
| ID | In-Distribution |
| OOD | Out-of-Distribution |
| PID | Proportional-Integral-Derivative |
| PPO | Proximal Policy Optimization |
| RGB | Red Green Blue |
| RL | Reinforcement Learning |
| RNN | Recurrent Neural Network |
| SAC | Soft Actor-Critic |
| SARSA | State-Action-Reward-State-Action |
| SE(3) | Special Euclidean Group (3D) |
| TD | Temporal-Difference |
| TD3 | Twin Delayed Deep Deterministic Policy Gradient |
| TRPO | Trust Region Policy Optimization |
| VAE | Variational Autoencoder |
| ViT | Vision Transformer |

VLA Vision-Language-Action
VLM Vision-Language Model
VQA Visual Question Answering
DTP Discretized Task Progress
MP Mean Progress

*We can only see a short distance ahead,*
*but we can see plenty there that needs to be done.*

— Alan Turing [94]

# 1 INTRODUCTION

## 1.1 MOTIVATION AND SCOPE

Robots operating in unstructured, changing environments often encounter conditions that differ from their training data. These Out-of-Distribution (OOD) environments pose a fundamental challenge: how can a robotic agent generalize its skills and adapt when faced with novel objects, lighting or task variations not seen before? Traditional robotics solutions, based on precise modeling and classical control, excel within well-modeled regimes but tend to break down under unmodeled dynamics or unforeseen situations [42]. Meanwhile, modern Artificial Intelligence (AI) approaches have achieved remarkable generalization in domains like vision and language by leveraging massive data and expressive models [7, 9, 11, 12, 21, 23, 41, 72, 91]. Scaling these systems to real-world robots has proven difficult: initial gains can deliver up to 90 percent reliability across various tasks [11], but the remaining 10 percent, comprised of rare edge cases that occur routinely in everyday environments, requires new techniques.

This thesis focuses on how to close that final gap by combining transformer-based foundation models with lightweight online adaptation on real hardware. We scope our discussion to three interrelated paradigms as promising paths toward robust OOD behavior: transformer-based foundation models (especially multimodal and VLA models), advanced RL techniques for continual adaptation, and hybrid architectures that combine learning-based methods with modular design inspired by neuroscience or classical frameworks. While Part I surveys all three paradigms to frame a broader research agenda, the empirical work in this thesis investigates RL-driven adaptation of a frozen VLA backbone on a physical SO-101 arm, with modular safety and control components inherited from the underlying LeRobot stack rather than evaluated in isolation.

A key design choice is to *not* fine-tune the weights of the large-scale VLA directly with RL. Prior work on RL fine-tuning of high-capacity visuomotor and diffusion-style policies has shown that naively updating the full model can induce catastrophic forgetting of valuable pre-trained representations unless large subsets of weights are frozen or low-rank adaptation layers are carefully constrained. In contrast, this thesis treats the VLA as a frozen visuomotor prior and attaches a residual RL head that learns an additive correction in action space. The two modules are coupled through an adapter-like interface in which the residual

parameters are trained independently of the backbone. This separation preserves the representations and instruction grounding learned during foundation-model pre-training while allowing the residual RL head to specialize behavior to the target embodiment and task distribution. RL is used where it is strongest - exploiting local experience to shape fine-grained motor behavior and improve robustness under distribution shift - without erasing the broad competence encoded in the base policy.

## 1.2 CONTRIBUTIONS AND OUTLINE

This thesis makes three primary contributions:

1. **A systematic, gap-driven literature synthesis.** We map transformer-based VLA models, online RL and modular control into one coherent landscape and identify where state-of-the-art systems fall short in OOD generalization.

2. **An end-to-end experimental pipeline for residual adaptation on real hardware.** We integrate a pretrained VLA prior (Isaac-Gr00t) with lightweight residual RL adapters and deploy the system on a 3D-printed SO-101 tabletop arm. The pipeline provides standardized data collection, safety-aware control, and RL training on real trajectories and is fully reproducible, with code to be released publicly upon thesis completion.

3. **Empirical evidence that residual RL can efficiently recover OOD performance.** We show that a lightweight residual RL head on top of a frozen VLA policy can recover a large fraction of performance lost under real-world distribution shift, using only tens of physical trials instead of retraining the full model.

The remainder of this thesis is structured around three parts. Part I reviews the surrounding literature and identifies the open problems that motivate the residual reinforcement learning approach used in this work. Part II outlines the experimental design, metrics, and implementation strategy that define the evaluation pipeline. Part III presents the empirical findings, including ablations, performance breakdowns, and error analyses that contextualize the strengths and limitations of the proposed method. Together, these parts form a coherent arc from motivation to methodology to empirical insight, guiding the reader through the conceptual foundations and practical contributions of the thesis.

# Part I

# REVIEW AND BACKGROUND

# 2

# ELEMENTS

This review does not attempt to catalogue every advance in robotic machine learning or its neighboring disciplines. Instead, it focuses on the core theoretical foundations and key paradigms that enable VLA models, RL, and classical robot control to interoperate.

## 2.1 CLASSICAL CONTROL THEORY

> "Every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it."
>
> John McCarthy, *Proposal for the Dartmouth Conference* (1955)

Classical control theory provides the mathematical foundations for regulating dynamical systems. Early work formalized the feedback control paradigm, exemplified by James Clerk Maxwell's analysis of governors in 1868 (ensuring stability of steam engine speed regulators). The field coalesced mid-20th century with rigorous stability criteria (e.g. Lyapunov functions) and optimal control formulations. A landmark contribution was dynamic programming that came from Bellman's principle of optimality, which introduced recursive value functions for optimal decision-making over time [5]. Bellman [5] showed that an optimal policy can be derived by backward induction on a value function, yielding what is now known as the Bellman equation. This paved the way for optimal control in continuous dynamics: Pontryagin's Maximum Principle (1962) provided first-order optimality conditions for control trajectories, and Kalman et al. [40] introduced the Linear Quadratic Regulator, giving closed-form optimal feedback for linear systems via solutions of the Riccati equation. Together, these developments established that controllability and observability (concepts first articulated by Kalman) are crucial system properties for design. Classical control also produced robust, easy-to-implement strategies like the PID controller with tuning rules from Ziegler and Nichols [107], which, though heuristic, became ubiquitous in engineering. In summary, classical control theory contributed a rich set of analytical tools (stability criteria, frequency-domain analysis, optimal control theory) that form the backbone for modern robotic control systems. These methods assume

known system models; challenges in modeling and high-dimensionality later motivated data-driven and learning-based approaches.

## 2.2 DEEP LEARNING

*Meaning lies as much in the structure of things as in their content.*

— DOUGLAS R. HOFSTADTER [36]

Alan Turing's 1950 essay posed the question "Can machines think?" and introduced the imitation game, later called the Turing Test, as an operational criterion for intelligence [94]. Early AI research embraced symbolic reasoning, producing search-based solvers, the Physical Symbol System Hypothesis [65] and rule-driven expert systems. Limitations in perception and scaling spurred the statistical turn: perceptrons [74], the universal approximation theorem [18], back-propagation [75] and probabilistic graphical models [67]. The deep learning era began when GPUs enabled AlexNet to win ImageNet 2012 [44], quickly followed by residual networks [34] and sequence models such as LSTM [35]. In reinforcement learning, Deep Q-Networks achieved human-level Atari play [62]. The transformer architecture [95] catalyzed large language models such as GPT-3 [13] and GPT-4, while multimodal pretraining produced models such as CLIP [71]. Today's state-of-the-art systems combine self-supervised pretraining, scaling laws and prompt-based adaptation, pushing AI toward generalist, vision-language-action agents.

### 2.2.1 *Representation Learning*

Deep learning must discover useful representations from data. For example, Figure 1 shows that classes are not separable in Cartesian coordinates $(x, y)$ but become linearly separable in polar coordinates $(r, \theta)$. While the mapping $f\colon (x, y) \mapsto (r, \theta)$ follows directly from coordinate geometry, a deep learning model must infer its own optimal transformation $f$ without relying on any predefined axioms.

The world's complexity could not be modeled linearly, often requiring non-linearity to capture complex relationships between concepts. For instance, separating foreground objects from camouflage backgrounds often requires a non-linear transformation such as a colour-opponent space (e.g. CIELAB [2]), which renders the decision boundary almost linear.

A common task of a supervised learning[1] system is to predict an output $\hat{\mathbf{y}} \in \mathbb{R}^k$ from some input $\mathbf{x} \in \mathbb{R}^d$. That process can be called $f$ where $\hat{\mathbf{y}} = f(x)$. With the following simple equation,

$$\hat{\mathbf{y}} = \sigma(W\mathbf{x} + \mathbf{b}), \tag{1}$$

1 Self-supervised and contrastive objectives have become the default pre-training paradigm for large-scale models, but supervised examples remain essential for calibration and evaluation.

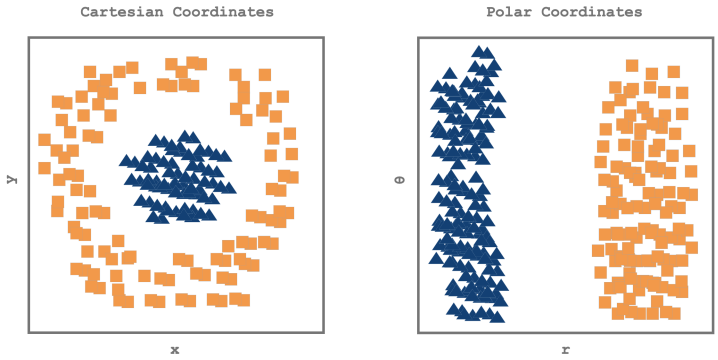



**Figure 1.** Illustration of representation learning: in Cartesian space the blue inner cluster and orange outer ring require a complex boundary, but in polar coordinates they separate linearly along the radial axis.

we can tweak the values of $W, b$ and $\sigma$ to produce accurate estimates of $\hat{y}$ if we have a valid training set to work with $\mathcal{D} = \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^d$ and $\mathbf{y}^{(i)} \in \mathbb{R}^k$. $W$ is weight, $\mathbf{b}$ is bias and $\sigma$ is a non-linear activation function such as ReLU or sigmoid.

Stacking multiple units of Eq. 1 together into $L$ layers provides the depth necessary for more complicated representations to emerge.

$$\mathbf{a}^{(\ell)} = \sigma(W^{(\ell)}\mathbf{a}^{(\ell-1)} + \mathbf{b}^{(\ell)}), \quad \ell = 1, \ldots, L. \tag{2}$$

The final prediction can be computed by running a forward-pass from $\mathbf{a}^{(0)} = \mathbf{x}$ all the way through $L$ layers to $\hat{\mathbf{y}} = \mathbf{a}^{(L)}$. The first prediction will not be very good since the set of weights $W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ have not been updated to bring the predictions as close as possible to the original training set $\mathcal{D}$. This can be solved by minimizing the loss between $y_i$ and $\hat{y}_i$ with a loss function

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)}).$$

Typically this is performed by a gradient-based optimization, namely gradient-descent where there parameters are updated via $\theta \leftarrow \theta - \eta\nabla_\theta J(\theta)$ where $\theta \in \{W^{(\ell)}, b^{(\ell)}\}_{\ell=1}^{L}$ and $\eta$ configures the *learning rate*.

Running this algorithm separately to improve the predictive $f$ capability is the machine learning; which becomes deep learning when $L > 2$.

The deeper the network the more complicated features it can learn. However the deeper it becomes the more parameters there are to train and the harder it is to *understand* what is actually going on inside the black box $f$.

Residual connections [34] enable gradients to bypass poorly conditioned layers, effectively learning a series of incremental $\Delta f$ refinements rather than an entire mapping from scratch. Formally, a residual block learns $\mathbf{y} = f(\mathbf{x}) + \mathbf{x}$, where $f$ is a small stack of convolutions and $\mathbf{x}$ is the identity shortcut. This simple reformulation mitigates vanish-

ing gradients and allows networks hundreds of layers deep to converge reliably.

### 2.2.2 *Convolutional Neural Networks*

Building on Eq. 2, neural networks can learn to predict outputs from a primitive vector-based dataset $\mathcal{D}$, but a useful extension is to give a capable network the ability to see in the real world.

Take an input (also called feature map) $X \in \mathbb{R}^{H\times W\times C}$ with $W, H$ and $C$ as the width, height and channel depth[2] of the image. Passing even a small RGB map $X \in \mathbb{R}^{100\times 100\times 3}$ through a fully connected layer would require 10,000 weights per neuron. This quickly becomes intractable for large enough feature maps so they must be compressed with minimal feature loss; a prime function of a Convolutional Neural Network (CNN).

The feature map is convolved with a kernel $K \in \mathbb{R}^{k_H\times k_W\times C\times C'}$, producing the $(i,j)$th compressed output as (where $i$ and $j$ correspond to individual pixels),

$$Y^{(\ell)}_{i,j,c} = (X * K)_{i,j,c} = \sum_{u=1}^{k_H}\sum_{v=1}^{k_W}\sum_{c=1}^{C} X^{(\ell-1)}_{i+u-1,\,j+v-1,\,c}\, K^{(\ell)}_{u,v,c',c}.$$

The final output of each layer with kernel $K^{(\ell)}$ and bias $b^{(\ell)}$is,

$$A^{(\ell)}_{i,j,c} = \sigma\left(Y^{(\ell)}_{i,j,c} + b^{(\ell)}\right),$$

where $\sigma$ is a non-linearity. Stacking $A^{(\ell)}$ gives the next layer's input of shape $(H^{(\ell+1)}, W^{(\ell+1)}, C^{(\ell+1)})$ which are depend on the padding $P$ and stride $S$ parameters,

$$H^{(\ell)} = \frac{H^{(\ell-1)} - k_H + 2P}{S} + 1, \quad W^{(\ell)} = \frac{W^{(\ell-1)} - k_w + 2P}{S} + 1.$$

POOLING. To reduce spatial size the input $X \in \mathbb{R}^{H\times W\times C_{\text{in}}}$ to something smaller, pixel regions are *pooled* into smaller subregions and passed to the next layer (red layers in Fig. 2). This is often taken over the max or average of a $p \times p$ patch,

$$P^{(\ell)}_{i,j} = \max_{0\le u,v<p} A^{(\ell)}_{ip+u,jp+v}, \quad \text{or} \quad P^{(\ell)}_{i,j} = \frac{1}{p^2}\sum_{0\le u,v<p} A^{(\ell)}_{ip+u,jp+v}.$$

FULLY CONNECTED LAYERS. Convolutions and pooling are performed until the feature map has been reduced enough to a fully connected layer learn the features.

The final feature map is flattened into a vector $\mathbf{h} \in \mathbb{R}^D$ where $D$ is the product of the dimensions of the last pooling layer (green lay-

2 An Red Green Blue (RGB) image has $C = 3$ for red, green and blue.

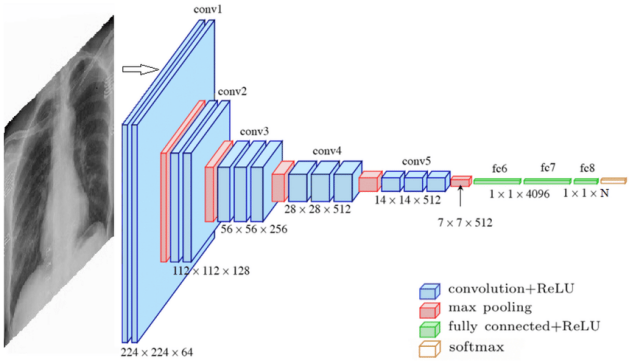



**Figure 2.** Architecture of a deep convolutional network showing five convolutional blocks (blue: conv+ReLU, red: max-pooling) that transform a $224 \times 224 \times 3$ input into a $7 \times 7 \times 512$ feature map, followed by three fully connected layers (green: ReLU, orange: softmax) producing an $N$-way classification. Adapted from Yang et al. [98].

ers in Fig. 2). Then final output is calculated using the softmax $\hat{\mathbf{y}} = \text{softmax}(W_{fc}\mathbf{h} + \mathbf{b}_{fc})$ which can be trained using $\theta \leftarrow \theta - \eta \nabla_\theta L(\theta)$.

The CNN was conceived by Fukushima [24], however it was LeCun et al. [47] that introduced end-to-end supervised learning with trainable convolutional filters via backpropagation, formalized weight sharing, and demonstrated state-of-the-art handwritten digit recognition.

VANISHING GRADIENTS. As networks grow deeper, gradients propagated through many layers tend to shrink toward zero, hindering effective weight updates in early layers. ResNets overcome this by adding identity skip connections,

$$A^{(\ell+1)} = f(A^{(\ell)}) + A^{(\ell)},$$

that let gradients flow unimpeded through shortcut paths. This simple modification enables training of ultra-deep CNN models without loss of signal [34].

COMPOUND SCALING. Scaling up CNNs to more layers can result in better accuracy as seen from ResNet-18 to ResNet-200 [34]. Although poorly understood, CNN scaling can occur across three dimensions: width [102], depth [34] or resolution scaling [38]. Tan and Le [90] formulated a unified scaling framework by using single compound coefficient $\phi$ to scale depth, width and resolution from their baseline values $d^{(0)} = L, w^{(0)} = W$ and $r^{(0)}$. Each increment of $\phi$ roughly doubles compute by enforcing $\alpha\,\beta^2\,\gamma^2 \approx 2$, where $\alpha, \beta, \gamma > 1$ are constants. The resulting dimensions are

$$(d, w, r) = \left(d^{(0)},\, w^{(0)},\, r^{(0)}\right) \odot \left(\alpha^{\phi},\, \beta^{\phi},\, \gamma^{\phi}\right).$$

### 2.2.3 *Autoencoders*

Autoencoders (AEs) are representation learning architectures that works in two parts: an encoder function $\mathbf{h} = f(\mathbf{x})$ and decoder that produces reconstruction $\mathbf{r} = g(\mathbf{h})$ with parameters $\{\theta_f, \theta_g\}$. It consists of a hidden layer $\mathbf{h}$ that describes the latent representation of the input to learn $g(f(\mathbf{x})) \approx \mathbf{x}$ by, minimizing a loss

$$\min_{\theta_f, \theta_g} L(\mathbf{x}, g(f(\mathbf{x})).$$

AEs come in many different forms, such as Sparse AEs $\min_{\theta_f,\theta_g} L(\mathbf{x}, g(f(\mathbf{x})) + \Omega(\mathbf{h})$ with a sparsity penalty $\Omega(\mathbf{h})$ for classification [25], Denoising AEs $\min_{\theta_f,\theta_g} L(\mathbf{x}, g(f(\tilde{\mathbf{x}}))$ with $\tilde{\mathbf{x}}$ being a corrupted copy of $\mathbf{x}$ for filtering noise [69] and others. Variational Autoencoders (VAEs) are of particular interest where instead of propagating signals through a latent bottleneck $\mathbf{h}$ they impose a probabilistic prior on the latent space,

$$\min_{\theta_f, \theta_g} \mathbb{E}_{f(\mathbf{h}|\mathbf{x})}[L(\mathbf{x},\, g(\mathbf{h}))] + D_{\mathrm{KL}}\Big(f(\mathbf{h} \mid \mathbf{x}) \,\|\, \mathcal{N}(0, I)\Big)$$

from $f(\mathbf{h} \mid \mathbf{x}) \sim \mathcal{N}(\mu, \sigma^2 I)$, which yields continuous, disentangled representations that can be smoothly sampled to generate novel data like images [15][3].

## 2.3 TRANSFORMERS

Sequence transduction modelling has been important for the fields of language translation and next token prediction,

$$(x_1, x_2, \ldots, x_T) \mapsto (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{T'})$$
$$(x_1, x_2, \ldots, x_T) \mapsto (\hat{y}_{T+1}),$$

where $x_t \in \mathcal{V}_x, y_t \in \mathcal{V}_y$. Recurrent Neural Networks (RNNs) like the Long Short-Term Memory (LSTM) proposed by Sutskever, Vinyals, and Le [88] showed early promise but quickly became difficult to scale since calculating a required hidden state $h_t = f(x_t, h_{t-1})$ depended on the previous state's value; no parallelization was possible.

With the introduction of the transformer by Vaswani et al. [95] paralleization was now possible with a new model architecture that calculated the self-attention between all tokens $(x_1, x_2, \ldots, x_T)$ in a sequence. Self-attention is computed by compiling three vectors: $\mathbf{k} = [k_1, k_2, \ldots, k_T]^T$ encoding the position of each token, $\mathbf{q} = [q_1, q_2, \ldots, q_T]^T$ the starting token, and $\mathbf{v} = [v_1, v_2, \ldots, v_T]^T$ a set of

3 The Kullback-Leibler (KL) divergence measures how a distribution $q(\mathbf{z})$ departs from a reference $p(\mathbf{z})$: $D_{\mathrm{KL}}(q \,\|\, p) = \int q(\mathbf{z}) \log\left(\frac{q(\mathbf{z})}{p(\mathbf{z})}\right) d\mathbf{z}$. It is always non-negative, equals 0 only when $q = p$, and is asymmetric [45].

value vectors that store the information content of each token[4]. Mathematically, attention is the weighted sum[5] from query vector $\mathbf{q}_k$,

$$\mathbf{z}_k = \text{Attention}(\mathbf{q}_k, \mathbf{k}, \mathbf{v}) = \sum_{t=0}^{T} \text{softmax}(\mathbf{q}_k \mathbf{k}_t^T)\mathbf{v}_t.$$

With the benefits of high parallezibility come from stacking multiple attention heads together into a multihead attention block,

$$\begin{aligned}\text{MultiHeadAttention}(\mathbf{q}_k, \mathbf{k}, \mathbf{v}) &= \text{concat}(\text{head}_1(\mathbf{q}_k), \ldots, \text{head}_h(\mathbf{q}_k)) \\ \text{where head}_i(\mathbf{q}_k) &= \text{Attention}(W_i^Q \mathbf{q}_k, W_i^K \mathbf{k}, W_i^V \mathbf{v}) W^O.\end{aligned}$$

Here the parameter matrices are $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{h\, d_v \times d_{\text{model}}}$.

This ability to capture long-range phenomena became noteworthy as language transduction models began to sound more human-like.

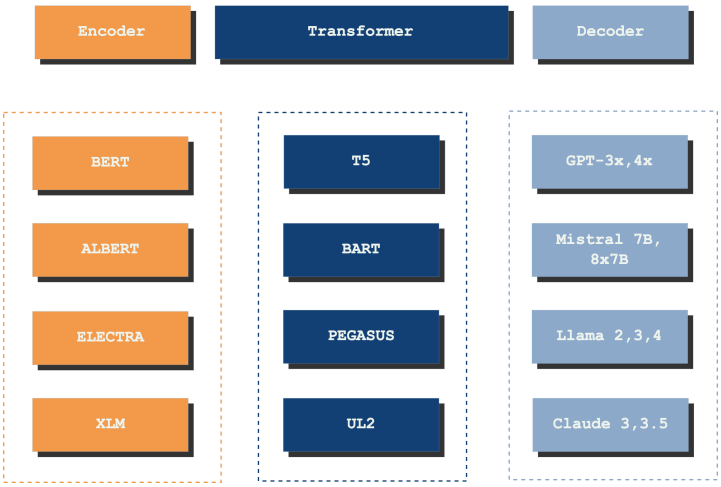



**Figure 3.** Overview of transformer model families grouped by architecture. *Left*: encoder only (BERT, ALBERT, ELECTRA, XLM). *Middle*: encoder-decoder (T5, BART, PEGASUS, UL2). *Right*: decoder only (GPT-3x/4x, Mistral 7B/8×7B, Llama 2/3/4, Claude 3/3.5).

### 2.3.1 *Large Language Models*

Different types of transformers have emerged most notable decoder-only [13], encoder-only [105] and a encoder-decoder mix [50] as shown in Fig. 3. Each has its advantages and disadvantages. Decoder-only transformers tend to perform better at open-ended sequence generation, code synthesis and in-context few-shot prompting due to their

4 In the literature $\mathbf{k}, \mathbf{q}$ and $\mathbf{v}$ are normally expressed as matrices $K, Q$ and $V$. Vectors were used here for exposition to highlight the per-token computation; in practice these are batched as matrices to enable efficient parallel processing.

5 The sum of the product $\mathbf{q}_k \cdot \mathbf{k}_t$ should be evenly diffused to avoid softmax generating one-hot vectors, where very large values in the softmax can drown out the smaller values. We don't want that. Hence it's safer to scale the dot product i.e., $\mathbf{q}_k \cdot \mathbf{k}_t / \sqrt{d_k}$ where $d_k$ is the attention head size [95].

autoregressive training. Encoder-only transformers perform better at discriminative tasks because of their Masked Language Model (MLM) bidirectional training.

Transformers, like any other neural network, can only take real numbers as inputs. Therefore translating language to numerical format, also known as tokenizing, can meaningfully dictate the capabilities of the language model. Nuemrous tokenizing schemes are employed today most notably WordPiece used by Bidirectional Encoder Representations from Transformers (BERT) [84] and Byte-Pair Encoding (BPE) used by Generative Pre-trained Transformer (GPT)-series [43]. Formally, a tokenizer is a mapping $T : \Sigma^* \to V^*$ from an input string over alphabet $\Sigma$ to a sequence of tokens in vocabulary $V$ (with $|V|$ fixed). For any word $w \in \Sigma^*$ we write $T(w) = (s_1, s_2, \ldots, s_k)$, where $s_i \in V$ and $w = s_1 \circ s_2 \circ \cdots \circ s_k$. Each token $s_i$ is then mapped to an integer ID via a lookup $f : V \to \{1, \ldots, |V|\}$.

BYTE-PAIR ENCODING. Start with $V_0 = \Sigma$ and counts $c_0(a, b)$ for every adjacent symbol pair $(a, b)$. At each merge step $t = 1, \ldots, N$, pick $(a^*, b^*) = \arg\max_{(a,b)} c_{t-1}(a, b)$, then define a new symbol $x = a^* b^*$ and update $V_t = V_{t-1} \cup \{x\}$ and counts by replacing all occurrences of $(a^*, b^*)$ with $x$. After $N = |V| - |\Sigma|$ merges, $V_N$ has the desired size. Tokenization greedily matches the longest symbol in $V_N$ at each step.

WORDPIECE. Given corpus $C$, learn $V$ by maximizing the data likelihood under a subword model. One often maximizes

$$\max_V \prod_{w \in C} \sum_{(s_1, \ldots, s_k) = w} \prod_{i=1}^{k} p(s_i \mid s_{<i}),$$

where $p(s_i \mid s_{<i})$ is estimated from subword counts. In practice a greedy heuristic adds the subword that most increases the log-likelihood until $|V|$ is reached.

### 2.3.2 *Vision Transformers*

While the Transformer architecture [95] has become the de-facto standard for natural language tasks, it has proven to be effective at computer vision tasks when paired with an effect tokenizer and a high data regime (14M-300M images) [20]. Propagating an image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ through a Vision Transformer (ViT), requires it to be flattened into a linear sequence of patches $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 C)}$ where $(P, P)$ is the resolution of each image patch and $N = HW/P^2$ is the resulting number of patches. Patches can then be treated equal to word tokens in a natural language context allowing the model to be trained on tasks like image classification in a supervised fashion. ViTs have shown remarkable generalization when trained on a large corpus of images, even outperforming CNNs [20].

### 2.3.3 *Vision Language Models*

Vision-Language Models (VLMs) extend the Transformer architecture to jointly model images and text. Modern VLMs fall into four broad families based on their pretraining objectives. Pioneering efforts such as Visual-BERT and ViLBERT embed image regions and words into a shared Transformer. They are trained on (a) masked token modelling to predict missing words or image regions, and (b) image–sentence matching to decide whether a caption describes an image [51, 87].

CONTRASTIVE–BASED VLM. Models like Contrastive Language-Image Pre-training (CLIP) train separate image and text encoders on hundreds of millions of image–caption pairs using an InfoNCE loss. Matched pairs are pulled together in a shared embedding space while mismatched pairs are pushed apart. At inference, text prompts can retrieve semantically aligned images in zero-shot fashion (e.g. CLIP with ResNet-101 attains 76.2% zero-shot ImageNet accuracy) [71].

SigLIP replaces CLIP's softmax-normalized contrastive loss with a pairwise sigmoid loss

$$L_{\text{sig}} = -\frac{1}{|\mathcal{B}|} \sum_{(i,j)\in\mathcal{B}} \Big[ y_{ij} \log \sigma(z_i \cdot z_j) + (1 - y_{ij}) \log(1 - \sigma(z_i \cdot z_j)) \Big],$$

operating per image–text pair without global normalization and yielding improved zero-shot accuracy and batch-size scalability [103].

MASKING–BASED VLM. Masking approaches view VLM pretraining as cross-modal denoising. FLAVA employs three Transformer encoders (image, text, multimodal) and combines unimodal/multimodal masked modelling with a contrastive loss, achieving state-of-the-art results on over 35 vision, language, and multimodal benchmarks after pretraining on 70 M image–text pairs [83]. MaskVLM extends this by masking raw pixels and text tokens simultaneously, using each modality to help reconstruct the other [46].

GENERATIVE–BASED VLM. Generative VLMs learn to produce text and/or images directly. CoCa adds a caption-generation loss atop contrastive training, enabling tasks like Visual Question Answering (VQA) without extra fusion modules; it is pretrained on over 1 B images with alt-text annotations [99]. Mixed-modal autoregressive models such as CM3Leon interleave image and text tokens in a single decoder, supporting both image-to-text and text-to-image generation with retrieval-augmented pretraining and supervised fine-tuning [100].

VLM FROM PRETRAINED BACKBONES. To reduce compute and data demands, many VLMs map frozen pretrained encoders into each other's space. Frozen maps visual features (NF-ResNet-50) into a frozen 7B-parameter Large Language Model (LLM), fine-tuning only a lightweight projection head on Conceptual Captions for rapid zero-

/few-shot multimodal capabilities [93]. MiniGPT-4 aligns BLIP-2's Q-Former+ViT image embeddings to a Vicuna LLM via a small projector trained on ∼5 M image–text pairs and an instruction-tuning step [106].

## 2.4 REINFORCEMENT LEARNING

*The most important thing for AI*
*is to have systems that can*
*learn from experience, like humans do.*

— YANN LECUN [108]

RL is a reward-based paradigm in which an agent improves its behavior through trial and error. Unlike supervised learning, which relies on labeled input–output pairs to minimize a predefined loss, and unsupervised learning, which uncovers structure without external feedback, RL uses scalar rewards to guide policy improvement over time.

### 2.4.1 *Formal Framework*

An RL problem is modeled as a Markov Decision Process (MDP) defined by a 5-tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$. The transition function is defined as $P : S \times A \to S$ and the reward function is $r : S \times A \to \mathbb{R}$. In general, for solving any agent-world RL task:

1. At each time $t$ the agents observe the current state $s_t \in \mathcal{S}$
2. Selects the appropriate action $a_t \in \mathcal{A}$ to perform
3. Environment returns a reward $r_t = r(s_t, a_t)$
4. Agent transitions to state $s_{t+1} = P(s_t, a_t)$

The goal should be to follow (and create) a policy $\pi : S \to A$ that maximizes the accumulated reward over time $V^\pi(s_t)$. That policy $\pi(a|s)$ maps states to action probabilities with the return from time $t$ as

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

with $\gamma \in [0, 1)$ as the discount factor applied to each successive reward $r_t$ determining whether the agent is myopic ($\gamma = 0$) and foresighted ($\gamma = 1$).

The value $v$ of a state $s$ under a policy $\pi$ is expressed as $V_\pi(s_t)$ or $V^\pi(s_t)$ depending on the textbook,

$$V^\pi(s) = \mathbb{E}_\pi\Big[G_t \mid s_t = s\Big],$$

and the action–value function is

$$Q^\pi(s, a) = \mathbb{E}_\pi\Big[G_t \mid s_t = s,\ a_t = a\Big].$$

The optimal policy $\pi^*$ then satisfies

$$\pi^*(s) = \arg\max_a Q^\pi(s, a).$$

The search for $a$ is the ultimate goal of any RL agent. There are two predominant methods: (1) if we know $Q(s, a)$ then solve for $a = \arg\max_a Q(s, a)$, referred to as value-based methods and (2) first find $\pi(s)$ to then sample $a \sim \pi(s)$, referred to as policy-based methods. These will be the focus of the remainder of §2.4.

### 2.4.2 *Value-Based Methods*

TD LEARNING. Temporal-Difference (TD) agents learn from experience over time (i.e., temporally). Modeling their impact on an environment as a function of the current state and action taken can provide enough stimulus to inform successive decisions,

$$V(s_t) \leftarrow V(s_t) + \underbrace{\alpha}_{\text{learning rate}} \underbrace{\left[ \underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma V(s_{t+1})}_{\text{discounted value of next state}} - V(s_t) \right]}_{\text{temporal difference}}.$$

An action $a_t$ that is taken in state $s_t$ at time step $t$ in accordance with policy $\pi$ is denoted as $Q_\pi(s_t, a_t)$ which is the action-value function. With the optimal policy $\pi^*$ one can arrive at the optimal action-value function,

$$Q^*(s_t, a_t) = \max_\pi Q^\pi(s_t, a_t).$$

Q-LEARNING. The agent maintains a table of action-value estimates, $Q(s, a)$, which it updates at each time step using the Bellman equation. This table records expected returns for every state–action pair and serves as a dynamic reference for decision making. Because updates are based on the highest-valued actions rather than the behavior policy $\pi$, this approach is classified as off-policy control [89].

In greedy action selection shown in Algorithm 1, the agent always chooses the highest-valued action, maximizing immediate reward but risking premature convergence to suboptimal behaviors. The $\varepsilon$-greedy policy mitigates this by selecting a random action with probability $\varepsilon$ (which decays over time) and otherwise exploiting the best known action; this ensures initial exploration of the state–action space while gradually shifting toward exploitation. The softmax policy offers a smoother trade-off by sampling actions according to a Boltzmann distribution over Q-values, controlled by a "temperature" $\tau$: high $\tau$ yields near-uniform exploration, low $\tau$ concentrates probability on the best actions. Thus, greedy maximizes exploitation, $\varepsilon$-greedy balances exploration and exploitation via occasional random moves, and softmax provides graded exploration proportional to relative action values.

**Algorithm 1** Q-Learning with Flexible Action-Selection

**Require:** learning rate $\alpha$, discount $\gamma$, episodes $M$
**Require:** decay factors $\lambda$ (for $\varepsilon$) and $\kappa$ (for $\tau$)
**Require:** `policyType` $\in$ {greedy, $\varepsilon$-greedy, softmax}

Initialise $Q(s,a) \leftarrow 0$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$
**for** episode $= 1$ to $M$ **do**
    $S \leftarrow$ initial state
    **for** $t = 0$ **until** $S$ is terminal **do**
        **if** `policyType` == `greedy` **then**
            $A \leftarrow \arg\max_a Q(S,a)$
        **else if** `policyType` == $\varepsilon$-`greedy` **then**
            $\varepsilon \leftarrow e^{-\lambda t}$
            Choose $A$ randomly with prob. $\varepsilon$ else $A \leftarrow \arg\max_a Q(S,a)$
        **else** ▷ softmax
            $\tau \leftarrow e^{-\kappa t}$
            Sample $A$ from $P(a) = \dfrac{\exp\Big(Q(S,a)/\tau\Big)}{\sum_i \exp\Big(Q(S,i)/\tau\Big)}$
        **end if**
        Take $A$, observe $R$ and next state $S'$
        $Q(S,A) \leftarrow Q(S,A) + \alpha\Big[R + \gamma \max_a Q(S',a) - Q(S,A)\Big]$
        $S \leftarrow S'$
    **end for**
**end for**

SARSA. In State-Action-Reward-State-Action (SARSA) an update $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$ depends on the action that the behaviour policy $\pi$ actually executes, so the algorithm's evaluation and control are fully coupled. With a purely greedy policy the agent always selects $A_{t+1} = \arg\max_a Q(S_{t+1}, a)$; SARSA then improves rapidly but can still inherit any risk that stems from stochastic transitions. Under a decaying $\varepsilon$-greedy policy the same update incorporates exploratory actions with probability $\varepsilon$, producing more conservative value estimates that often yield safer paths than those found by off-policy Q-learning. If $\pi$ is a softmax policy, the contribution of $Q(S_{t+1}, A_{t+1})$ is weighted by a Boltzmann distribution whose temperature $\tau$ gradually cools, allowing the algorithm to hedge among near-optimal actions before committing. Thus greedy SARSA maximises exploitation, $\varepsilon$-greedy SARSA balances exploration and exploitation explicitly, and softmax SARSA provides a graded, uncertainty-aware compromise; each choice trading convergence speed against robustness to sub-optimal early estimates.

### 2.4.3 *Policy-Based Methods*

POLICY GRADIENT. The goal of policy learning is to directly optimize $\pi_\theta(a|s) \sim \mathcal{N}(\mu, \sigma)$ such that $\theta \in \mathbb{R}^d$ to sample the next best action $a \sim \pi(s)$. This can be achieved my maximizing the expected (discounted) return,

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]. \tag{3}$$

Computing the gradient of Eq. 3, $\nabla_\theta J(\theta)$ using the policy gradient theorem yields,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) Q^\pi(s_t, a_t)\right].$$

Gradient ascent can be used to the optimal $\pi^*(s)$.

VARIANCE REDUCTION. The above gradient calculation incorporates high amounts of variance, so modifications can be made to ensure training is more stable with the choice of $\Psi_t$ in,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \Psi_t \nabla_\theta \log \pi_\theta(a_t|s_t)\right],$$

from the following [78],

1. $\sum_{t=0}^{\infty} r_t$ total reward for the trajectory
2. $\sum_{t'=t}^{\infty} r_{t'}$ reward following action $a_t$
3. $\sum_{t'=t}^{\infty} r_{t'} - b(s_t)$ baselined version of (2)
4. $Q^\pi(s_t, a_t) = \mathbb{E}_{\substack{s_{t+1:\infty}, \\ a_{t+1:\infty}}}[\sum_{l=0}^{\infty} r_{t+l}]$ state-action value function

5. $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$ advantage function
6. $r_t + V^\pi(s_{t+1}) - V^\pi(s_t)$ TD residual where $V^\pi(s_t) = \mathbb{E}_{\substack{s_{t+1:\infty}, \\ a_{t:\infty}}} [\sum_{l=0}^{\infty} r_{t+l}]$

Once a concrete value for $\nabla_\theta J(\theta)$ is selected, gradient updates are straightforward,

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta).$$

### 2.4.4 *Hybrid Methods*

ACTOR-CRITIC

*Give me six hours to chop down a tree and*
*I will spend the first four sharpening the axe.*

— Abraham Lincoln

Actor–critic methods integrate policy-gradient (actor) and value-function (critic) ideas in a single on-policy loop. The actor, parameterized by $\theta$, selects actions and is updated with the critic's feedback, while the critic, parameterized by $w$, learns a low-variance estimate of either $V_w(s)$ or $Q_w(s, a)$ as described in detail in Algorithm 2. The TD error $\delta_t$ couples the two, yielding faster, more stable convergence than pure policy-gradient and better asymptotic performance than value-only control.

**Algorithm 2** On-Policy Actor–Critic (Q-form)

Initialise $s$, $\theta$, $w$ randomly
Sample $a \sim \pi_\theta(a \mid s)$
**for** $t = 1$ to $T$ **do**
    Sample reward $r_t \sim R(s, a)$ and next state $s' \sim P(s' \mid s, a)$
    Sample next action $a' \sim \pi_\theta(a' \mid s')$ ▷ on-policy
    $\theta \leftarrow \theta + \alpha_\theta \, Q_w(s, a) \, \nabla_\theta \log \pi_\theta(a \mid s)$
    $\delta_t \leftarrow r_t + \gamma \, Q_w(s', a') - Q_w(s, a)$
    $w \leftarrow w + \alpha_w \, \delta_t \, \nabla_w Q_w(s, a)$
    $s \leftarrow s', \quad a \leftarrow a'$
**end for**

TRUST REGION OPTIMIZATION. A policy parameter update step, $\theta \leftarrow \theta + \alpha\eta$ creates a new policy from the old counterpart $\pi_{\theta_{\text{old}}}$. If the gradient update is stable i.e., $\eta$ does not blow up then the ratio of the two should be relatively low.

Trust Region Policy Optimization (TRPO) enforces this using a KL divergence constraint,

$$\mathbb{E}[\mathrm{KL}(\pi_{\theta_{\text{old}}}(.\,|s) \| \pi_\theta(.\,|s))] \leq \delta, \quad D_{\mathrm{KL}}(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)} \, dx.$$

where $\delta \in \mathbb{R}$ is a hyperparameter [77].

PROXIMAL POLICY OPTIMIZATION. Proximal Policy Optimization (PPO) replaces TRPO's second-order trust-region constraint with a first-order penalty that is easier to compute yet preserves stability [79]. Given an old policy $\pi_{\theta_{\text{old}}}$ and an updated policy $\pi_\theta$, define the probability ratio

$$r(\theta) = \frac{\pi_\theta(a \mid s)}{\pi_{\theta_{\text{old}}}(a \mid s)}.$$

If $\hat{A}_{\theta_{\text{old}}}(s,a)$ is a high-quality advantage estimate, vanilla policy-gradient would maximize

$$J^{\text{TRPO}}(\theta) = \mathbb{E}_{\pi_{\theta_{\text{old}}}}\Big[r(\theta)\,\hat{A}_{\theta_{\text{old}}}(s,a)\Big].$$

Large deviations of $r(\theta)$ from unity, however, can cause destructive updates. PPO therefore clips the ratio to the interval $[1-\epsilon,\,1+\epsilon]$ with a hyperparameter $\epsilon$, yielding the clipped surrogate

$$J^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\Big(r_t(\theta)\,\hat{A}_t,\ \text{clip}\Big(r_t(\theta), 1-\epsilon, 1+\epsilon\Big)\hat{A}_t\Big)\Big].$$

If the sign of $\hat{A}_t$ is positive, the clip caps overly aggressive improvements; if negative, it limits harmful drops in probability. The objective thus enforces a soft trust region without computing KL divergences, Hessian–vector products, or line searches.

For practical training the surrogate is combined with a squared-error value loss and an entropy bonus that encourages exploration:

$$J^{\text{CLIP}'}(\theta) = J^{\text{CLIP}}(\theta) - c_1\Big(V_\theta(s) - V_{\text{target}}\Big)^2 + c_2\,H\Big(s, \pi_\theta(\cdot)\Big),$$

where $V_\theta$ is the critic, $H$ is policy entropy, and $c_1, c_2$ balance the three terms. Optimization proceeds with first-order stochastic gradient ascent using multiple epochs and minibatches drawn from the same trajectory batch. Empirically, PPO matches or exceeds TRPO's performance while being simpler to implement and significantly faster per update.

### 2.4.5 *Methodological Considerations*

Value-based algorithms learn quickly with low variance because each transition updates many state–action estimates, yet they rely on an explicit maximisation step that becomes expensive or brittle in large or continuous action spaces. Policy-based algorithms sidestep this maximisation and naturally support continuous or stochastic behaviours, but their gradient estimates contain high variance and therefore consume far more samples. Hybrid actor–critic schemes exploit the strengths of each approach: a critic supplies low-variance value targets to guide the actor, while the actor eliminates the costly arg-max. This compromise tends to deliver steadier learning and better scalability for real-world robotics and control tasks, where action spaces are high-dimensional and data collection is costly.

## 2.5 EMBODIED INTELLIGENCE AND AGI

> *It is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one year old when it comes to perception and mobility.*
>
> — HANS MORAVEC [64]

Modern consensus holds that intelligence emerges from the closed action loop between an agent and its environment. Embodiment theory therefore argues that a robot must *experience* the world through its own sensorimotor contingencies to acquire transferable concepts. VLAs offer rich perceptual priors, while RL supplies the machinery for continual interaction-driven learning; together they form a plausible path toward embodied Artificial General Intelligence (AGI). Throughout the review we emphasize how this triad-vision, language and action-grounds abstract reasoning in physical competence, a prerequisite for safe autonomous robots operating in open worlds.

# 3 ROBOTIC FOUNDATION MODELS

The term "foundation model" in AI refers to large models (often trained on vast data from the internet) that provide a general base which can be adapted to many downstream tasks. In robotics, foundation models are emerging as a way to achieve generalist behavior across many tasks and environments, leveraging multimodal understanding. Two lines of work converged here: (a) multi-task learning in robotics, where a single network is trained to perform many different tasks (across possibly different embodiments), and (b) vision-language models that exhibit general-purpose understanding and reasoning. The combination leads to VLA models; policies that ingest visual observations and language instructions, and output robotic actions.

## 3.1 TRANSFORMER FOUNDATION MODELS

The early work on Gato demonstrated transformer-based generalism across multiple domains. Gato was trained on 604 distinct tasks spanning vision, language, and control, from captioning images to playing Atari games and controlling a real robot arm [72]. With a single set of weights, Gato could accomplish diverse tasks, including robotic manipulations. For instance, on a real-robot block stacking benchmark, Gato could stack objects of novel shapes with success rates comparable to a specialized imitation-learning policy [72]. This suggests that large transformers can learn a multi-task prior that generalizes to some new variations. However, Gato's performance also underscored limitations: while it mastered many training tasks, solving an entirely new task outside its training distribution still required fine-tuning or additional data [72].

Transformers have also been applied to narrow domains such as planning. SayCan used an LLM to interpret high-level instructions and suggest possible robot actions, which were then executed by low-level policies [1]. SayCan demonstrated robots responding to commands like "bring me a water bottle" by decomposing them, but the approach was modular (LLM + separate learned skills) rather than end-to-end.

PaLM-E (Embodied PaLM) is another broad model, embedding sensor observations into a language model context, enabling a form of LLM-planning for robotics (e.g. reasoning about visual inputs and issuing actions) [21]. These broader models extend the concept of a "robotic foundation model" beyond just direct visuomotor policies, towards multimodal reasoning agents. The theoretical promise is an embodied general intelligence that flexibly applies knowledge to new scenarios, yet practically they reveal how far current models must go to truly generalize in the real world.

A landmark was Robotics Transformer 1 (RT-1), which collected 130,000 demonstrations across 700+ household tasks for a mobile manipulator and trained a Transformer policy that takes image and textual goal description, and outputs motor commands [11]. RT-1 achieved robust multi-task performance (e.g. picking up arbitrary objects specified by language) within its training distribution.

## 3.2 VLA FOUNDATION MODELS

### 3.2.1 *Architecture*

Recent breakthroughs introduce VLA models as large-scale "foundation" policies that integrate vision and language understanding to directly generate low-level robot actions. By leveraging pre-trained vision-language knowledge from the web and massive multi-task training, these models aim to provide generalist skills out-of-the-box and zero-shot adaptation to new instructions, drastically reducing the need for task-specific programming or extensive new demonstrations.

VLAs extend the capabilities of VLMs and LLMs by incorporating an action-generation component, as illustrated in Fig. 4. These models unify perception and instruction understanding into an end-to-end system that produces control outputs, enabling robots to act on internet-scale knowledge with minimal task-specific engineering.

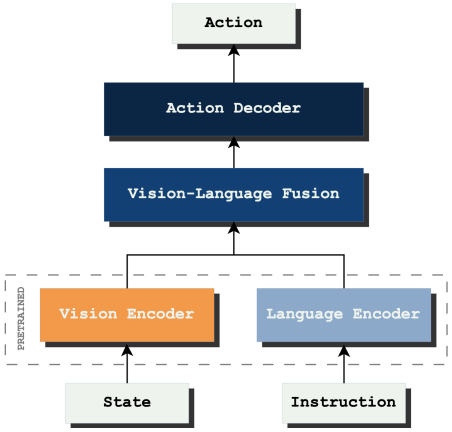



**Figure 4.** VLA architecture. A pretrained vision encoder ingests the robot's state and a pretrained language encoder processes the instruction. Their outputs feed into a shared action decoder, which generates the final action command.

The current landscape of VLA models employ a variety of visual encoder backbones, namely CNNs [6, 54, 58, 59, 82, 92], ResNets [17, 26–28, 32, 39, 55, 72, 81, 104], EfficientNets [11, 16, 29], ViTs [4, 7, 8, 37, 41, 52, 66, 80, 86, 101], and hybrid ViT-ResNets [73] to compress visual input into a feature $v \in \mathbb{R}^{d_v}$. Each leverages different inductive biases, offering a spectrum of trade-offs between representational richness and computational efficiency [56]. ViTs have exhibited superior accuracy relative to CNN architectures under data-rich regimes, while requiring substantially less compute to train [20].

Visual encoding is only one half of the encoding process. Language encoders often use either a GPT decoder-only [26–28, 32, 37, 52, 55, 73, 80–82, 101], BERT encoder-only [58, 59] or encoder-decoder [4, 11, 12, 16, 39, 66, 86, 92] architecture to compress language input into a feature $\ell \in \mathbb{R}^{d_\ell}$. Relative performance of GPT and BERT-style models varies across different tasks e.g., paraphrasing vs predicting [105].

The vision language fusion module then computes a joint representation $f = \psi(v, \ell) \in \mathbb{R}^{d_f}$ via cross-attention or Feature-wise Linear Modulation (FiLM) [68] for an action-decoder to produce a distribution over actions by

$$\pi(a|f) = \text{softmax}\left(Wf + b\right),$$

where $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, $W \in \mathbb{R}^{|\mathcal{A}| \times d_f}$ and $b \in \mathbb{R}^{|\mathcal{A}|}$. The final action output might include 6-Degrees of Freedom (DoF) Special Euclidean Group (3D) (SE(3)) end-effector velocities $a = [\dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]$, a binary gripper command [12] and the mean and covariance of the uncertainty in the velocities $a \sim \mathcal{N}(\mu_f, \Sigma_f)$ [37, 52].

### 3.2.2 *Comparative Analysis*

Building on the architectural overview above, we now compare five representative VLA systems across shared design axes and real-world benchmarks. RT-2 integrates a ViT visual encoder with a GPT-style decoder and softmax action head, yielding 62% zero-shot success on novel-object pick-and-place tasks, an improvement from its predecessor RT-1 [11] driven by web-scale vision pretraining [12]. Gemini-ER combines a ViT backbone and encoder–decoder language model, achieving a two-fold gain on generalization benchmarks by leveraging dual-stage fine-tuning across Internet and robot data [91]. NVIDIA's GR00T N1 emphasizes data efficiency: minimal on-robot tuning on a bimanual humanoid platform delivers 76.8% average success, reflecting its lightweight ResNet-FiLM fusion [7]. Figure AI's Helix adopts separate slow (7 Hz) reasoning and fast (200 Hz) visuomotor loops, enabling high-frequency control of thousands of novel objects and robots [23]. Physical Intelligence's $\pi_0$ model added a flow-matching policy head atop a pre-trained vision-language backbone, training on the largest cross-embodiment dataset to date and generating high-frequency continuous trajectories for 68 tasks over eight robotic platforms [8]. Its successor, $\pi_{0.5}$, incorporated 400 hours of teleoperated data from 100 environments and multiple embodiments, demonstrating superior real-world generalization by co-training on heterogeneous dataset [9]. While these results showcase the power of diverse encoder–fusion–head pipelines, direct comparisons as shown in Table 1 remain challenging due to inconsistent benchmarks and reporting. Standardized ablations of fusion strategies and unified evaluation suites are needed to identify which architectural choices most strongly drive data efficiency, generalization and control precision.

Fig. 5 highlights control frequency as a critical axis for real-time, continuous motor control. Models with large vision-language backbones such as PaLM-E and Gemini-ER often struggle to sustain fast inference

rates and require special hardware to run, limiting their applicability in tasks requiring sub-millisecond feedback in embedded systems [91]. A dual-system design mitigates this bottleneck by decoupling high-level reasoning from low-level control: a lightweight, high-frequency (100–500 Hz) policy handles continuous actions, while the VLM operates at slower rates to manage complex perception and planning. This separation underpins the modularity axis, allowing each subsystem to be optimized for its specific temporal demands and ensuring both rapid response and sophisticated interpretation.

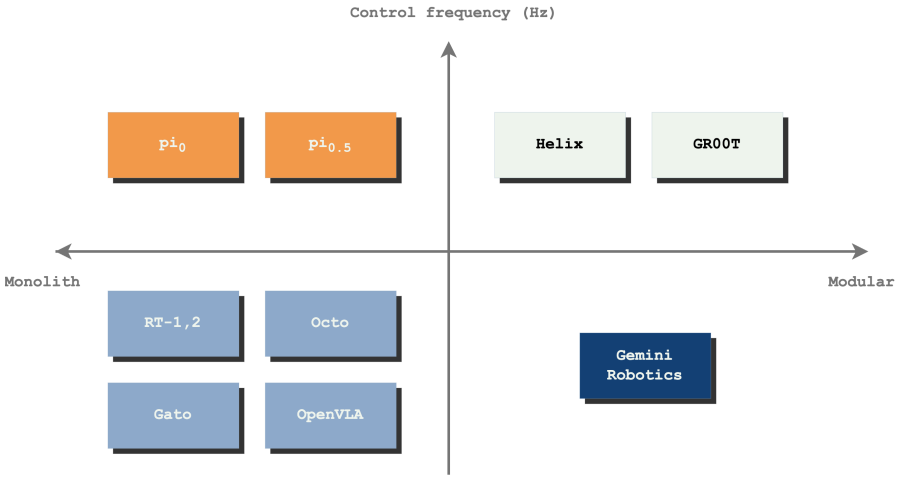



**Figure 5.** Taxonomy of Robotic Foundation Models along the axes of Modularity (horizontal) and Control Frequency (vertical). Models in the upper half exhibit higher embodiment coupling (e.g., proprioceptive inputs, closed-loop control), while those on the right are more modular, decoupling perception, reasoning, and control.

**Table 1.** Real-world task success rates for VLA models. These average success rates were measured under diverse evaluation protocols and do not capture important nuances in real-world benchmarking. Differences in task definitions, completion criteria, environment variability, sensor noise, hardware configurations and quality-assurance procedures mean that values are not strictly comparable across models. Readers should interpret these rates in the context of each model's specific testing methodology.

| MODEL | AVG. SUCCESS RATE (%) | MODEL SIZE | TRAINING (STEPS) |
|---|---|---|---|
| Gato [72] | 50.0[a] | 1.18B | 1M |
| RT-1 [11] | 97.0[b] | 35.0M | 130K |
| Octo [92] | 72.0[c] | 93.0M | – |
| RT-2 [12] | 62.0[d] | 55.0B[i] | 80K[i] |
| Gemini Robotics [91] | 65.0[e] | –[*] | – |
| OpenVLA [41] | 70.6 | 7.00B | 970K |
| GR00T N1 [7] | 76.8[f] | 2.00B | – |
| $\pi_0$ [8] | 78.0[g] | 3.30B[k] | 993M[l] |
| $\pi_{0.5}$ [9] | 82.5[h] | 3.30B | 993M+ |

[*] Not disclosed but rumored to be 1T+ parameters.
[a] ∼50% success on block-stacking of novel objects (5 held-out shapes, 200 trials each) after fine-tuning; on par with single-task expert.
[b] ∼97% on seen training tasks (700+ instructions) and ∼76% zero-shot generalization to unseen instructions; tested on Everyday Robots fleet with distractors and novel backgrounds.
[c] ∼72% average across six manipulation scenarios (e.g. cable insertion, coffee making, bimanual tasks) after few-shot fine-tuning; ∼29% higher than RT-1-X zero-shot and matched RT-2-X on shared tasks.
[d] Measured on novel and unseen tasks using real-robot deployments.
[e] Multi-task average across pick-and-place and industrial assembly on a bimanual humanoid platform.
[f] Evaluated with few-shot demos on folding, handling, wiping.
[g] Consolidated average across pre-training demos for stacking bowls, laundry, bussing and dryer unloading.
[h] Similar to (g) but for dishes in sink, items in drawer, laundry and making bed.
[i] 55B PaLI-X variant trained over 80K steps and 12B PaLM-E variant over 1M steps.
[k] 3.3B = 3.0B VLM + 300M action.
[l] 993M = 903M from robot demos plus 90M steps from open datasets and task-specific fine-tuning of 1–10h per task.

### 3.2.3 *Real-World Benchmarks*

To measure progress towards general-purpose, real-world-capable robots, we require benchmarks that test both the breadth and adaptability of robotic intelligence. This section surveys existing benchmarks and proposes criteria for evaluating integrated RL-VLA systems, with emphasis on real-world performance.

MULTI-TASK AND GENERALIZATION BENCHMARKS. Early benchmarks for generalist robots have emerged in simulation. Meta-World is a suite of 50 manipulation tasks (e.g., push button, open door) used to evaluate multi-task learning and meta-learning algorithms. A single policy's ability to master all Meta-World tasks (and generalize to

new ones) is an indicator of general-purpose skill. Another is RLBench, a library of 100+ simulated tasks with diverse tool use and settings, which provides a playground for language-conditioned policies and task generalization. Results from models like $\pi_0$ and Gato on these benchmarks show decent *in-distribution* performance and some ability to adapt to withheld tasks [8, 72]. For example, Gato was evaluated on DM Control Suite tasks and Atari games alongside robotic control, and scaling laws were studied to see performance trends across domains [72]. However, simulation benchmarks, while useful, often don't capture the noise and variability of the real world.

REAL-WORLD PERFORMANCE BENCHMARKS. A few benchmarks explicitly target real robot generalization. One notable example is the RGB-Stacking Benchmark introduced by Lee et al. [49], where a robot arm must learn to stack novel shapes of blocks by training on some shapes and testing on unseen ones. Gato's evaluation on RGB-Stacking's Skill Generalization task showed it could match a specialist policy's success rate around ∼50% [72], providing a quantitative measure of cross-generalization. There are also household task benchmarks, for instance, the BEHAVIOR benchmark (Benchmark for Everyday Household Activities in Virtual, 2021) defines 100 activities (like cleaning a table, making coffee) in simulation but with an eye towards real-world complexity; it has a concept of generalization to new object instances and environments [85]. Some researchers have attempted these tasks on real robots like the Toyota HSR or Stretch, evaluating success in real home settings. Another emerging evaluation is open-vocabulary object manipulation: tests where a robot is asked to pick or arrange objects specified by category or attribute (possibly unseen during training). RT-2 and Helix were implicitly tested on this e.g., "pick up the strawberry toy" vs "pick up the dinosaur toy" where the latter might be novel. Embodiment transfer could also be considered a benchmark: how well a policy trained on one robot does on another with minimal adjustment (Gemini's embodiment adaptation is a case study). A crucial piece for real-world benchmarking is measuring robustness and safety: for example, the "Real-World RL" suite [22] proposes metrics like how often policies fail catastrophically, how much human intervention is needed, etc. For integrated RL+VLA, we might want a benchmark that requires an agent to understand an instruction, then learn a new skill to fulfil it. Imagine a test where a robot is told, "Here is a new tool, figure out how to use it to hang this picture," success would require comprehension (VLA) and on-the-spot learning (RL). While no standard benchmark yet captures this full spectrum, individual components exist that can be combined.

EVALUATION CRITERIA AND RESEARCH GAPS IN BENCHMARKING. Current benchmarks often address either generalization *or* learning, but integrated systems need both. Evaluation criteria should include: (a) Generalization to novel objects/tasks (test on things not in training data), (b) Sample efficiency/Adaptation (measure how

many trials or demos needed to learn a new task to a certain performance level), (c) Multi-modality (the agent must handle vision, language and possibly other sensors), and (d) Long-horizon performance (tasks requiring many steps, where planning or subgoals are needed). A combination of simulated and real benchmarks is ideal; simulation for rapid testing and ablation, and real for ultimate validation. One gap is the lack of a widely-adopted real-world benchmark suite for generalist robots. This is partly because deploying identical evaluations on physical robots is hard (equipment differences, etc.). Some initiatives, like the REAL Competition [3] or Roboturk/Robomimic datasets [57], push toward standardizing evaluation by providing common environments or datasets. Benchmarking RL-VLA integration might involve tasks like: follow a novel instruction that requires learning a new behavior (e.g., "Learn to use this screwdriver to tighten a bolt" where "using a screwdriver" was not in training). Success would be measured by the agent's ability to figure it out within a limited number of tries. As research progresses, the community may define challenge tasks akin to a Robotic IQ test that spans reasoning, memory, and learning. For now, researchers rely on assembling evidence across multiple benchmarks: if a new model beats baselines on Meta-World, RGB-Stacking, and perhaps a real kitchen task, this is considered progress. In conclusion, developing better benchmarks is itself an open problem, necessary to drive the field toward truly general-purpose robotic intelligence.

## 3.3 GENERALIZATION IN REAL-WORLD ENVIRONMENTS: CURRENT STATE AND GAPS

While foundation models show impressive generalist capabilities in lab settings, their ability to handle the open-ended complexity of the real world is limited. This section examines the state-of-the-art in real-world capable transformer-based robotic learning and identifies gaps in generalization and online learning.

ROBUSTNESS AND DISTRIBUTION SHIFT. Transformer-based policies often face a reality gap when moving from curated training data to unstructured real environments. Even the best VLA models can struggle with OOD scenarios i.e., situations not seen during training. For instance, a model trained on indoor tabletop manipulation might falter in a completely new setting (say, outdoors or in a cluttered home with novel object types). Generalization is typically tested by holding out some object types or task variations. As noted, Gato's real-robot evaluation involved stacking objects of unseen shapes, where it achieved moderate success [72].

Similarly, RT-2 was demonstrated to handle some novel objects and instructions beyond its direct robot experience. However, these cases remain relatively constrained (novel combinations of familiar elements) [12]. Truly entirely unseen environments, with new lighting conditions, unfamiliar object categories, or unforeseen physics remain a tough chal-

lenge. The current state is that no transformer-based robotic agent can yet be deployed to any arbitrary real-world environment with reliable performance. Safety and reliability concerns also emerge: models might misinterpret unusual inputs, leading to unsafe actions. A research gap here is how to make these large models more robust to distribution shift, possibly through better simulation-to-real transfer, domain randomization, or online adaptation.

LACK OF ONLINE LEARNING AND ADAPTATION. Most transformer-based policies are trained offline and then frozen (or only lightly fine-tuned) before deployment. In contrast, human and animal intelligence continuously learns from experience in real time. Current systems have little ability to improve on the fly when faced with new challenges. For example, if a robot encounters a tool it has never seen, a fixed model can only rely on its prior knowledge (which might be insufficient), whereas an ideal agent would experiment and learn about the tool. Online RL (or other forms of continual learning) are generally absent from VLA pipelines. This is a significant gap: without online learning, models cannot systematically improve in new environments or recover from errors by refining their policy. Some initial efforts are exploring this e.g., using RL fine-tuning to improve pretrained VLA models [31] but integrating real-time learning remains an open research area. Challenges include catastrophic forgetting (learning new things without overwriting old knowledge) and safety during exploration. Thus, a crucial research direction is how to imbue transformer-based agents with online adaptation, possibly through hybrid architectures or constrained exploration that keeps the robot and its environment safe.

SAMPLE EFFICIENCY AND REAL-WORLD DATA CONSTRAINTS. In real-world robotics, collecting data is expensive and time-consuming, which makes sample efficiency pivotal. Yet many transformer-based approaches demand huge datasets. VLA models like RT-2, Gemini-ER and $\pi_o$ are enabled by internet-scale pretraining or thousands of human demonstrations [8, 12, 91]. This raises concerns: can these models generalize without such extensive data for every new situation? Current evidence suggests diminishing returns when encountering scenarios not covered by training data. For instance, Gemini Robotics needed additional fine-tuning to specialize in long-horizon tasks or new embodiments, implying that pretraining alone was insufficient for those capabilities. Moreover, models may not generalize to new embodiments (robot hardware) without retraining because the dynamics and kinematics differ. While Gemini reports some success in adapting to new robots via fine-tuning [91], true embodiment-agnostic control remains unsolved. In summary, the current state-of-the-art transformer models mark substantial progress toward real-world robotics, but fall short in generalizing to truly novel situations, learning continuously, and minimizing data requirements. The following sections delve into specific limitations and complementary approaches addressing these issues.

LOW-LATENCY CONTROL. Dynamic environments demand not only robust policies but also rapid inference to close the perception-action loop. For smooth locomotion and manipulation, end-to-end latency (from sensor capture to motor command) must often fall within 2–10 ms or 100–500 Hz using $f = 1/T$, to match human-like responsiveness [30]. Higher latencies introduce perceptible lag, degrading control stability and reducing the agent's ability to react to unforeseen perturbations. Without real-time inference, even a well-trained model cannot generalize effectively in fast-changing settings; objects move, lighting shifts, and contacts occur unpredictably. Thus, designing transformer-based architectures or deployment pipelines that guarantee low-latency execution (via model compression, distillation, or hardware-aware optimizations) is essential for dependable real-world performance.

# 4

# ROBOTIC REINFORCEMENT LEARNING

> "It is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility."
>
> Hans Moravec [64]

Building off the theoretical foundation developed in §2.4 now with focus on applying to RL specifically to real robotic policies. RL provides a complementary perspective to the largely offline-trained VLA models. This section explores state-of-the-art RL approaches in robotics, focusing on policy gradient methods and hybrid techniques, and briefly contrasts them with older, sample-inefficient algorithms.

## 4.1 POLICY-GRADIENT METHODS

### 4.1.1 *Actor-Critic*

Policy gradient algorithms have become a cornerstone for continuous control in robotics. PPO is a prime example, it's a model-free RL method known for its stability and reliability in training complex policies [31, 79]. PPO and related actor-critic methods (like Asynchronous Advantage Actor-Critic (A3C)/Advantage Actor-Critic (A2C), Deep Deterministic Policy Gradient (DDPG), Soft Actor-Critic (SAC)) strike a balance between improving the policy and not deviating too far from the current policy (which can destabilize learning) [33, 53, 60]. In practice, PPO has been used successfully in simulated robotics tasks and even on hardware for learning locomotion or manipulation, often achieving state-of-the-art results on benchmarks in terms of final performance. The theoretical appeal of policy gradient methods is that they directly optimize for the expected reward, which is aligned with the task goal, and can naturally handle continuous action spaces.

Methodologically, PPO introduces a clipped objective that limits how much a policy can change per update, addressing the variance and collapse issues of earlier policy gradient attempts. In robotics, PPO and SAC are frequently used as baseline algorithms that provide strong performance given enough training time [33, 79]. Key findings include their ability to learn robust behaviors (like grasping or walking) in simulation. However, even these advanced methods often require thousands to

millions of environment interactions to converge, which is a challenge for real-world use. Nonetheless, they represent the state-of-the-art RL toolkit that any new robotic learning approach must consider, either as a competitor or a component.

## 4.2 HYBRID RL APPROACHES

Modern robotics research often blends pure RL with other strategies to improve efficiency and effectiveness. One hybrid approach is rewarded imitation learning or initializing RL with demonstration data (sometimes called warm-start or behavioral cloning initialization). This can dramatically cut down exploration time by starting the agent near a good policy. Another approach is hierarchical RL, where a high-level policy (could be learned or hand-designed) breaks tasks into subtasks that low-level controllers execute. This can be seen as a coarse form of integration between symbolic planning and RL. PPO itself can be considered hybrid in the sense that it leverages a critic (value function) to guide the policy updates (actor), combining value-based and policy-based methods [79]. There are also model-based RL methods, which learn a model of the environment and plan or generate imagined experience from it. These have shown higher sample efficiency on some tasks but can struggle when learned models are inaccurate. A recent trend is combining learning-based policies with search or planning at runtime for example, using a library of learned skills and a high-level search (like in the SayCan paradigm [1], where an LLM chooses which skill to apply). In terms of algorithmic innovation, soft policy gradient methods (like SAC [33]) maintain a maximum entropy objective that encourages exploratory yet stable behavior, which has improved learning speed and robustness. The literature also explores sim-to-real transfer (domain randomization, fine-tuning in real) as a hybrid between pure learning and leveraging simulated experience. Key takeaway: The cutting edge in RL for robotics is not one algorithm, but combinations of learning strategies that address exploration, credit assignment, and transfer, often built on the backbone of strong policy gradient methods like PPO [79].

## 4.3 GENERALIZATION IN REAL-WORLD ENVIRONMENTS: CURRENT STATE AND GAPS

SAMPLE-INEFFICIENCY. It's instructive to contrast the above with earlier generations of RL that proved impractical for robotics due to sample inefficiency. Classic Q-learning and value-iteration methods (without function approximation) were limited to very small state spaces [96]. The advent of Deep Q-Network (DQN) showed that neural nets could learn value functions from high-dimensional inputs, but algorithms like DQN (and its variants) still required enormous numbers of steps (e.g., millions of frames on Atari) to learn, and extended poorly to continuous control [61]. Early policy gradient algorithms like REIN-

FORCE had high variance and typically needed careful tuning and baseline subtraction to work [97]. Methods like Evolutionary Strategies (ES) or grid search over policies were even more sample-inefficient, though simpler [76]. In robotics, brute-force methods (like random search in policy space, or PILCO which attempted Gaussian process models) were limited by data requirements or model inaccuracies [19]. These outdated techniques are referenced for historical context: they underscore why the field pivoted to using function approximators (deep nets) and why on-policy learning alone is tough when each trial is costly. For instance, a robot hand learning to rotate an object with naive RL could take hundreds of hours of practice, which is infeasible without simulation. Modern methods like PPO improved stability but still rely on many trials [79]; thus, much of the research has been about improving sample efficiency (via off-policy algorithms, better exploration, or leveraging prior data). Overall, the field has largely moved past pure trial-and-error learning towards more data-efficient hybrids, but the lessons from those older methods (the need for better exploration, the danger of unstable updates) inform today's algorithm design.

# Part II

# METHOD

# 5 RESEARCH SYNTHESIS

Transformer-based VLA models have ushered robotics into an era of true quasi-generalist capability, demonstrating that a single model can, to an extent, see, reason and act across diverse tasks; yet leading examples [7, 9, 23, 91], despite enabling zero-shot manipulation of novel objects, remain far from the master-of-all-trades that real-world deployment demands. By weaving together the threads of classical control, AI, transformer-based foundation models, RL and modular architectures, we synthesize these advances to shape our research approach, leveraging breadth from pre-trained models, depth from online adaptation and structure from modular design, and we highlight the gaps that must be bridged for truly adaptive robotic learning in OOD environments.

## 5.1 KEY INSIGHTS

**FOUNDATION MODELS ENABLE BREADTH.** Foundation models provide robots with a priori knowledge and generalization capabilities previously unseen in robotics. Models like RT-2 and PaLM-E demonstrate that a single model can leverage web-scale data to perform many tasks and even exhibit zero-shot behaviors [12, 21]. This breakthrough allows handling *novel inputs* (new objects or instructions) because the model likely encountered similar examples during training (e.g., RT-2 recognizing a "coffee mug" it never saw in fine-tuning). However, foundation models alone are not a panacea: they tend to be heavy, inflexible after training, and their real-world performance can degrade under distribution shift without further adaptation [70].

**RL ENABLES DEPTH.** RL gives robots the ability to improve with experience in new scenarios. RL can fine-tune a policy to dynamics or reward specifics not covered in pre-training [42]. It is essential for *online learning*: when encountering an OOD environment, RL can adjust the policy incrementally. Yet RL alone is too slow and unsafe to tackle large distribution shifts from scratch. The synergy is clear: a foundation model supplies a strong initial policy, and RL refines that policy to the specific context efficiently [79], achieving *fast adaptation* with orders of magnitude fewer trials than learning from zero.

**HYBRID AND MODULAR ARCHITECTURES PROVIDE A STRUCTURAL PRIOR.** By not relying on a single monolithic neural network, hybrid and modular architectures enforce safety, handle multi-objective trade-offs, and incorporate specialist knowledge. For example, a classical Proportional-Integral-Derivative (PID) loop can maintain low-level stability, and safety checks can override learned policies if a collision is imminent. Structured models often outperform end-to-

end approaches in robustness [22]. Moreover, modular design aligns with practical engineering: individual modules (e.g., vision systems) can be upgraded without retraining the entire policy, which is vital for long-term system development.

RESEARCH HYPOTHESIS. An approach that integrates a transformer-based foundation model (for generalization), RL (for adaptation), and a modular architecture (for safety and structure) is a promising path to adaptive robotic learning. Alone, each component has limitations: foundation models without adaptation may fail under distribution shift, RL without a strong prior is too slow, and rigid modular systems lack novelty handling. Together, they form a more complete solution.

From these insights, our research hypothesis is reinforced: an approach that integrates a transformer-based foundation model (for generalization), RL (for adaptation), and a modular architecture (for safety and structure) is a promising path to adaptive robotic learning. None of these components alone would likely succeed: foundation models without adaptation might fail when their assumptions break, RL without a strong prior is too slow, and a rigid modular system without learning can't handle novelty. Together, they form a more complete solution.

## 5.2 OUTSTANDING GAPS AND CHALLENGES

GENERALIZATION VS. SPECIALIZATION BALANCE. Large foundation models excel at broad generalization but can lack the precision or efficiency of smaller, specialized policies trained via RL for specific robots and tasks. To bridge this gap, we must develop methods such as task-specific fine-tuning of large models or distilling their knowledge into compact high-frequency controllers that preserve broad capabilities while delivering task-level performance. For example, one could distill RT-2's policy into a lightweight network optimized for real-time execution. Our research will investigate fine-tuning and distillation strategies that adapt foundation models to particular robots and environments without sacrificing their general knowledge.

SAFETY AND TRUSTWORTHINESS. Modular architectures allow embedding safety rules, yet formally verifying that a learning-enabled robot remains safe under all real-world variations is generally intractable. Control-theoretic tools (e.g., Lyapunov stability analysis) offer partial guarantees, but neural policies defy exhaustive analysis. Practical approaches include safety shields and extensive simulation under perturbations-practices we will adopt. A critical open problem is online monitoring: detecting when the robot enters an unknown regime and gracefully reverting to a provably safe baseline. We will explore anomaly-detection modules (e.g., flagging out-of-distribution sensor inputs via a generative model) to trigger safe mode, acknowledging this remains a challenging research frontier.

CONTINUAL LEARNING AND MEMORY. While our current framework enables fast adaptation in a new environment, it does not ensure retention of prior skills. As a robot encounters ever more varied settings, it should accumulate competencies without catastrophic forgetting. True continual learning is unsolved: naively fine-tuning via RL will overwrite earlier knowledge. Potential solutions include modular compartmentalization of new skills (e.g., adding task-specific subnetworks) or meta-learning schemes that consolidate parameters. We identify the stability–plasticity dilemma: maintaining performance on environment A while learning B, as a key gap. We plan experiments to measure forgetting and to test methods such as Elastic Weight Consolidation or orthogonal gradient descent for mitigating it.

BENCHMARKING AND EVALUATION. There is currently no standard benchmark for "adaptive robotics." In this thesis we take a first step by introducing a structured real-robot evaluation protocol, but more broadly we argue that the community should develop a comprehensive evaluation suite drawing on modified Meta-World tasks, Colosseum-style real-world perturbation tests, and sensor-failure scenarios to measure adaptation across sensor, environment, and task changes. Such a protocol, spanning new objects, varied terrain, and shifting goals, would provide breadth and consistency and could serve as a shared benchmark for fair comparison.

## 5.3 SYNTHESIS FOR APPROACH

Considering all of the above, we synthesize the approach as follows: We will start with a strong foundation (a pretrained transformer model endowed with multimodal understanding) and anchor it to a reliable control backbone (ensuring baseline competence and safety). On this foundation, we will layer an adaptation mechanism (RL fine-tuning or online learning) that refines performance in novel scenarios. The expected outcome is a robot that, upon encountering an OOD situation, initially behaves reasonably (not randomly, thanks to the foundation model) and then improves rapidly through experience (thanks to RL), all while never venturing into catastrophic behavior (thanks to safety modules and constraints).

By iteratively testing this system on increasingly challenging distribution shifts, we will validate the core hypothesis. Success will be measured by the robot's ability to maintain high task success with minimal drop after a shift and how quickly it can recover any lost performance. For example, if introducing a 30% perturbation in the weight of an object causes the success to drop from 90% to 50%, can our system learn to return, say, 80% + success in a small number of trials? Does it avoid dangerous attempts during that adjustment?

The literature gives optimism that each piece can achieve parts of these goals: foundation models have shown the ability to do things like pick up unfamiliar objects correctly on the first try, and meta-RL has

shown fast improvement within tens of trials on new tasks in simulation. The novel contribution of our work will be to seamlessly integrate these pieces in real robotic hardware and demonstrate end-to-end that the robot can take what it "knows" (from pre-training) and adapt what it does not (through experience), in a unified system.

In the next section, we transition from what needs to be done to how we will do it. We outline the concrete research questions we will answer and the experimental methodologies, metrics, and implementation steps that will operationalize this synthesized approach.

# 6 METHODOLOGY

## 6.1 FORMULATED RESEARCH QUESTIONS

Drawing on the synthesis from §5, we set out four focused research questions that shape the experimental design. Each question links a concrete objective with a testable hypothesis and identifies the key methodological lever or research gap it addresses.

1. **RQ1: To what extent does a transformer-based foundation model pre-trained on broad data generalize to novel robotic tasks and unseen environments?**

   *Objective*: Quantify zero-shot and few-shot success, failure modes and transfer gaps across varied manipulation tasks.

   *Hypothesis*: The model will handle moderate perceptual changes yet falter when action dynamics or task semantics shift significantly.

2. **RQ2: Can online RL fine-tuning restore or surpass baseline performance after large distribution shifts within tens of physical trials, and which fine-tuning strategy is most efficient?**

   *Objective*: Compare full-model versus partial-layer updates, on-policy versus off-policy data and safety-aware reward design. Record trials-to-recovery and asymptotic performance.

   *Hypothesis*: With a strong initialization from the foundation model and possibly demonstration data as a warm start, RL adaptation will achieve substantial improvement in tens of episodes rather than thousands, representing a significant gain in sample efficiency.

3. **RQ3: Do modular architectures that divide fast low-level control from slower high-level reasoning yield more robust real-time behavior than monolithic policies?**

   *Objective*: Benchmark a two-tier controller (embedded "cerebellum" at 200–500 Hz, cloud planner at 1–10 Hz) against an end-to-end baseline under latency, sensor noise and obstacle perturbations.

   *Hypothesis*: The modular design will sustain task success under latency stress whereas the monolith will degrade.

4. **RQ4: What communication abstraction between modules best preserves global optimality while allowing independent retraining of sub-networks?**

*Objective*: Evaluate interfaces such as shared latent spaces, explicit symbolic messages and probabilistic belief states in simulated pick-and-place and long-horizon assembly tasks. Measure learning stability, sample efficiency and retrain cost.

*Hypothesis*: A compact stochastic latent protocol aligned with information-bottleneck theory will minimize retrain overhead while maintaining overall performance.

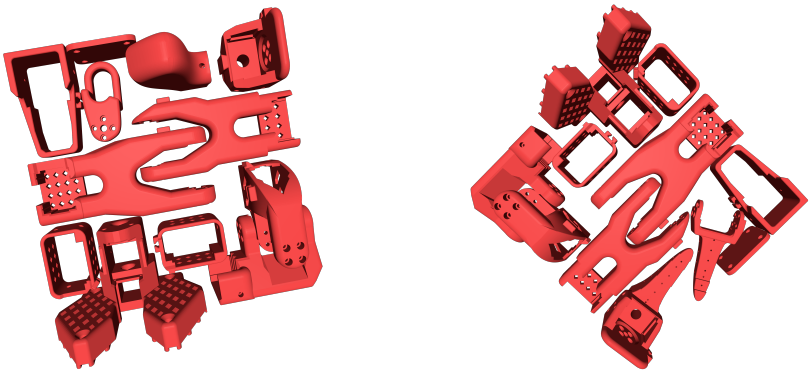
This honours thesis investigates **RQ2** within the available timeframe. RQ1, RQ3 and RQ4 define the broader research landscape and motivate future work.

## 6.2 EXPERIMENTAL PLATFORM

### 6.2.1 *Robotic Hardware*

The robotic hardware used for all real-world experiments is summarized in the tables in this section and illustrated by the STL layouts in Fig. 6, the assembled SO-101 manipulators in Fig. 7, and the complete tabletop setup in Fig. 8. For real-world experiments, a physical SO-101 3D-printed robotic arm was used as the manipulator, as shown in Fig. 7 . The arm is a teleoperated tabletop platform with three rotational degrees of freedom and a parallel gripper, rigidly mounted to a flat work surface. Visual observations of the workspace were provided by fixed external RGB cameras positioned to capture both the robot and the task area. All control, data recording, and policy inference were performed on a desktop computer equipped with a consumer-grade NVIDIA RTX 3080 and RTX 5080 Graphics Processing Units (GPUs). Variations in the tabletop environment and object set across tasks introduced distribution shifts in object geometry and mass, surface friction, clutter, and camera viewpoint, providing a range of real-world conditions for evaluation.

*A complete material list can be found in* **??**.



**(a)** Leader robot components **(b)** Follower robot components

**Figure 6.** STL layouts of the leader and follower robot components arranged on the Ender 3 printer bed for fabrication.

**Figure 7.** A pair of white 3D-printed SO-101 robotic arms with black ST-3215-C0xx servos, resting side-by-side on a clean white surface.

### 6.2.2 *Sensing and Vision System*

The sensing and vision system was designed to provide visually simple, low-clutter observations to the visuomotor backbone and residual RL adapters. Experiments were conducted in a predominantly white office environment with minimal background texture, including whiteboard panels placed directly behind the main manipulator to create a uniform backdrop, as visible in Figs. 7 and 8. This configuration reduces background clutter and high frequency visual noise, which can otherwise induce spurious correlations when policies are trained directly on pixel observations and lead to brittle behaviour under small appearance changes.

The primary camera configuration consists of a top camera mounted on a tripod that provides an overhead view of the workspace and a front facing camera attached to the desk divider, as illustrated in Fig. 8. The front camera captures lateral motions of the SO-101 arm and interactions across the horizontal extent of the task space, while the top camera supplies a bird's eye view that localises both the arm and the objects on the table with minimal occlusion. Multiple alternative camera placements were evaluated, including single front views and oblique side views. These alternatives frequently suffered from occlusions when the arm or gripper passed between the camera and the objects, which degraded grasp accuracy and increased the rate of failed manipulations.

*All cameras were running at the recommended 30 FPS [14].*

An auxiliary camera, also shown in Fig. 8, is used for task monitoring, reward computation, and partitioning the workspace into discrete zones to curate balanced and diverse training data, as illustrated in Fig. 9. Separating this monitoring stream from the main observation views allows the same physically fixed camera geometry to serve both control

**Figure 8.** Experimental tabletop manipulation setup with a Lenovo top camera mounted on a tripod providing an overhead view of the white workspace, a front facing Microsoft LifeCam attached to the desk divider, and a Logitech auxiliary camera used for task monitoring, reward computation, and partitioning the workspace into discrete zones to curate balanced and diverse training data. A 3D printed SO-101 robotic arm and gripper operate on a colored training cube and a Rubik's cube on the central table, with whiteboards, an office chair, and USB cables routing connectivity between cameras and the control machine.

and data curation, while maintaining a consistent, noise reduced visual input for policy learning and evaluation.

### 6.2.3 *Computing Infrastructure*

All real time inference for Isaac-Gr00t and the residual RL adapters runs on a local Linux workstation equipped with NVIDIA RTX 3080 and RTX 5080 GPUs. Initial experiments on the RTX 3080 (10 GB VRAM) led to frequent crashes and unstable robot behaviour, as Isaac-Gr00t alone required at least 8 GB of VRAM and left insufficient headroom for system and logging processes as displayed in Table 2. As a result, all final real robot experiments were executed on the RTX 5080, which provided enough memory headroom to ensure stable policy execution and reliable teleoperation. The workstation also hosted the LeRobot

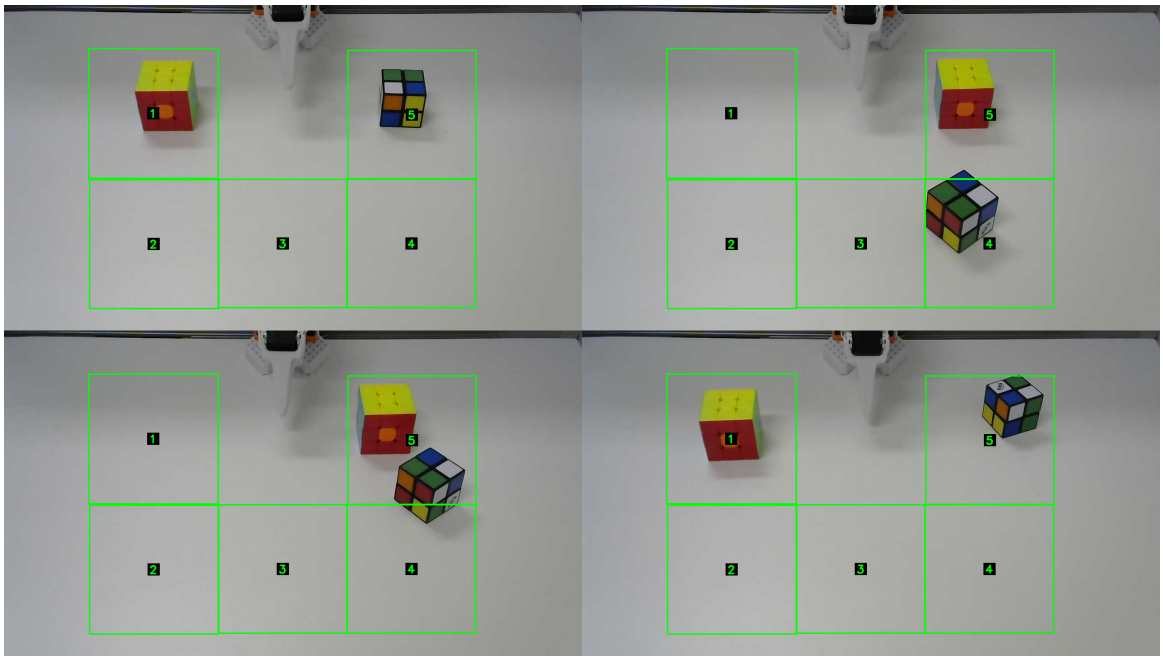



**Figure 9.** Top down view from the auxiliary monitoring camera with the workspace partitioned into a discrete grid using opencv [10]. The zone indices are used to balance data collection across object positions and interaction regions for reward computation and dataset curation.

control stack and data logging pipeline, recording synchronized camera streams, proprioceptive states, and actions for analysis.

**Table 2.** GPU VRAM, devices per job, and indicative cloud cost used in this work.

| GPU MODEL | VRAM | GPU PER JOB | COST PER JOB HR (USD) |
|---|---|---|---|
| RTX 3080[a] | 10 GB | 1 | 0.00[b] |
| RTX 5080 | 16 GB | 1 | 0.00[b] |
| RTX 4090 | 24 GB | 1 | 0.00[b] |
| A100 | 40 GB | 2 | 0.06996 |

[a] Isaac Gr00t requires ≥8 GB of VRAM for inference. The 10 GB capacity of the RTX 3080 operates near saturation when accounting for OS overhead and auxiliary processes.
[b] Zero marginal cost incurred as these devices were pre-owned by the author or affiliates.

All learning of residual policies and value functions was performed remotely on Modal Cloud [63]. Training jobs were launched via the Modal SDK (see Listing 1) and used two NVIDIA A100 GPUs per run, enabling efficient batched rollouts and gradient updates over the collected SO-101 trajectories. This separation of concerns placed computationally intensive training on scalable cloud hardware while reserving the local workstation for low latency control and high fidelity data capture on the physical robot.

**Listing 1.** Minimal Modal SDK entrypoint that launches a two A100 cloud training run from a single `modal run train.py` command.

```
@app.function(gpu="A100:2")
def train():
  # training logic
  pass

@app.local_entrypoint()
def main():
  train.remote()
```

### 6.2.4 *Datasets and Pretraining*

Datasets are collected exclusively on the physical SO-101 3D printed arm using the HuggingFace LeRobot platform, which provides a unified interface for logging observations, actions, and task metadata. No third party robot datasets are incorporated, in order to avoid mismatches in camera geometry, sensing quality, and embodiment between this setup and external collections. For each manipulation task, a set of teleoperated demonstrations is recorded and then augmented with additional on robot rollouts generated during online learning. All trajectories share a consistent, calibrated camera configuration for the SO-101 workspace, together with proprioceptive state and low level control commands.

The visuomotor backbone is the pretrained Isaac-Gr00t policy, used in its publicly released configuration. No additional pretraining of Isaac-Gr00t is performed in this work. Learning is instead restricted to lightweight residual adapters and value functions that are trained on top of the frozen backbone using the task specific trajectories described above. Perception and instruction grounding capabilities therefore originate from Isaac-Gr00t's upstream pretraining, while the collected SO-101 dataset is used solely to specialise behaviour to the target hardware and task distribution.

### 6.2.5 *Control Architecture and Software Stack*

The control architecture is built on top of the LeRobot framework [48], which provides a unified software stack for kinematics, low level control, and data logging for the SO-101 arm. High level components specify desired end effector poses or joint targets, while LeRobot handles conversion to hardware compatible commands and enforces joint limits and velocity constraints before streaming actions to the embedded controllers.

Geometric reasoning is implemented through a dedicated `RobotKinematics` module that wraps the `placo` library for both forward and inverse kinematics, as illustrated in Listing 2. The solver loads the SO-101 URDF, fixes the base, and defines a frame task at the `gripper_frame_link`. Forward kinematics takes joint angles in de-

grees, converts them to radians, updates the internal robot model, and returns a 4x4 transformation of the end effector in the world frame. This representation is used for logging, visualization, and defining Cartesian targets.

**Listing 2.** Skeleton of the `RobotKinematics` helper used in the LeRobot stack. Forward and inverse kinematics run in the same URDF joint space used for control and joint limit enforcement.

```
class RobotKinematics:
    def __init__(self, urdf_path: str,
                 target_frame_name: str = "
                    gripper_frame_link"):
        import placo, numpy as np
        self.robot = placo.RobotWrapper(urdf_path)
        self.solver = placo.KinematicsSolver(self.
            robot)
        self.solver.mask_fbase(True)
        self.joint_names = list(self.robot.
            joint_names())
        self.tip_frame = self.solver.add_frame_task(
            target_frame_name, np.eye(4)
        )

    def forward_kinematics(self, q_deg):
        import numpy as np
        q_rad = np.deg2rad(q_deg[: len(self.
            joint_names)])
        for name, q in zip(self.joint_names, q_rad):
            self.robot.set_joint(name, q)
        self.robot.update_kinematics()
        return self.robot.get_T_world_frame("
            gripper_frame_link")

    def inverse_kinematics(self, q_deg, T_target):
        import numpy as np
        q_rad = np.deg2rad(q_deg[: len(self.
            joint_names)])
        for name, q in zip(self.joint_names, q_rad):
            self.robot.set_joint(name, q)
        self.tip_frame.T_world_frame = T_target
        self.solver.solve(True)
        self.robot.update_kinematics()
        q_sol = [self.robot.get_joint(n) for n in
            self.joint_names]
        return np.rad2deg(q_sol)
```

Inverse kinematics receives the current joint configuration and a desired 4x4 end effector pose, then uses `placo`'s `KinematicsSolver` to compute a new joint configuration. Position and orientation weights allow trading off strict pose tracking against more flexible solutions that prioritize positional accuracy over orientation when needed. The resulting joint angles are converted back to degrees, and any additional

gripper joints are preserved from the input vector. In practice, this kinematics layer enables LeRobot policies and teleoperation commands that are expressed in task space to be executed as smooth joint space trajectories on the physical arm.

### 6.2.6 *Teleoperation Interface and Data Collection Pipeline*

Demonstration data are collected using LeRobot's leader follower teleoperation interface for the SO-101 arm. As shown in Listing 3, a single command launches a process that configures one SO-101 as the leader device and a second SO-101 as the follower. The leader arm streams its measured joint angles over serial in real time, and the follower arm tracks these targets with low latency, reproducing the demonstrated motion in the main workspace.

During teleoperation, LeRobot records synchronized trajectories that include joint positions, gripper state, and derived end effector poses from the kinematics module, along with timestamps. When camera streams are active, RGB frames are captured and time aligned with the robot state, producing pixel based demonstrations suitable for visuomotor learning. The resulting trajectories are written in LeRobot's dataset v3 format, organized into episodes that can be directly consumed by the subsequent behavior cloning and residual reinforcement learning pipelines without additional conversion.

**Listing 3.** Command used to launch the SO101 leader follower teleoperation pair for data collection. The follower arm receives low latency joint targets from the leader device over serial, enabling real world demonstration recording.

```
python -m lerobot.scripts.teleoperate \
    --robot.type=so101_follower \
    --robot.port=/dev/ttyACM1 \
    --robot.id=so101_follower_arm \
    --teleop.type=so101_leader \
    --teleop.port=/dev/ttyACM0 \
    --teleop.id=so101_leader_arm
```

### 6.2.7 *Safety Protocols*

Safety constraints are implemented in the LeRobot action processing pipeline. Before any data collection or reinforcement learning, joint limits are calibrated using the built in `lerobot-find-joint-limits` utility shown in Listing 4.

The end effector processor in Listing 5 clips Cartesian poses to a bounded workspace and limits per step translations, while joint space utilities cap relative motor targets (`max_relative_target`), gripper commands are clipped to a safe range, the motor bus disables torque on disconnect, and reset routines interpolate trajectories in small steps.

**Listing 4.** Command used to identify per-joint limits for the SO101 follower arm. The resulting configuration file is reused at training and deployment time to clip actions to a safe workspace.

```
lerobot-find-joint-limits \
  --robot.type=so101_follower \
  --robot.port=/dev/ttyACM1 \
  --robot.id=so101_follower_arm \
  --teleop.type=so101_leader \
  --teleop.port=/dev/ttyACM0 \
  --teleop.id=so101_leader_arm \
  --config_path=src/lerobot/configs/joint_limit_
     config.json
```

Together, these mechanisms provide a simple safety layer around teleoperation and RL control on the SO-101 arm.

**Listing 5.** End effector safety processor in the LeRobot stack. The step clips the commanded pose to a bounded workspace and limits the maximum translation between consecutive commands.

```
@dataclass
class EEBoundsAndSafety(RobotActionProcessorStep):
    end_effector_bounds: dict
    max_ee_step_m: float = 0.05
    _last_pos: np.ndarray | None = field(default=None
        , init=False, repr=False)

    def action(self, action: RobotAction) ->
        RobotAction:
        # current target position in Cartesian space
        pos = action.ee_pose[:3, 3]

        # clip to workspace bounds
        pos = np.clip(
            pos,
            self.end_effector_bounds["min"],
            self.end_effector_bounds["max"],
        )

        # limit per step motion to max_ee_step_m
        if self._last_pos is not None:
            dpos = pos - self._last_pos
            n = float(np.linalg.norm(dpos))
            if n > self.max_ee_step_m and n > 0:
                scale = self.max_ee_step_m / n
                pos = self._last_pos + dpos * scale

        self._last_pos = pos
        action.ee_pose[:3, 3] = pos
        return action
```

## 6.3 EVALUATION METRICS

To rigorously evaluate the generalization capabilities of the ResRL system compared to Base and RL baselines, this study moves beyond binary success rates to employ a fine-grained analysis of task execution. The evaluation framework centers on the DTP metric, which quantifies partial success to isolate specific failure modes across perception, kinematics, and dynamics.

### 6.3.1 *Primary Metric: Discretized Task Progress (DTP)*

Standard binary metrics often obscure the underlying causes of failure in robotic manipulation. A failure to grasp an object (control error) is fundamentally different from a failure to locate it (perception error), yet both result in a zero-success score. To resolve this ambiguity, a continuous progress score $P_{\text{task}} \in [0, 1]$ is assigned to every trial instead

of using $P_{\text{task}} \in \{0, 1\}$ [8]. This enables the measurement of the semantic stage reached by the policy.

As defined in Table 3, each task is segmented into four semantic milestones. These scores are aggregated to calculate the MP across $N$ trials,

$$MP = \frac{1}{N} \sum_{i=1}^{N} P_{\text{task}}^{(i)}, \tag{4}$$

where a higher $MP$ indicates robust partial completion and helps diagnose the specific bottleneck of a given policy.

**Table 3.** Definitions of discretized task progress ($P_{\text{task}}$) milestones. This framework distinguishes between early perceptual failures and late-stage manipulation failures.

| Score | Semantic Stage | Definition | Testing Focus |
|---|---|---|---|
| 0.00 | Detection Failure | The robot fails to approach the object or moves to an incorrect location. | Vision (VLA) |
| 0.25 | Approach/Contact | The end-effector makes physical contact with the target object but fails to initiate a stable grasp. | Calibration/IK |
| 0.50 | Grasp Stability | The object is successfully lifted $>$ 5cm off the surface but is dropped during transport. | Dynamics (RL) |
| 0.75 | Transport | The object is transported to within 5cm of the target zone but placement fails (for example, collision). | Control (RL) |
| 1.00 | Task Success | The object is placed in the target state and the robot retreats successfully. | End-to-End |

### 6.3.2 *Trial Protocol: The 5–5–10 Split*

For each combination of *Method* (BASE, RL, RESRL) and *Task*, the policy is evaluated with a fixed budget of 20 real-world rollouts, indexed $i = 1, \ldots, 20$. Throughout, the method labels are written in small caps: BASE denotes the supervised base policy trained on offline demonstrations, RL denotes the RL baseline trained from scratch without access to BASE, and RESRL denotes the residual RL adapter composed with a frozen copy of the base policy. In RESRL, RL updates only the residual head; the base policy parameters are not modified.

A budget of 20 trials balances statistical reliability with experimental feasibility. Fewer rollouts would make it difficult to distinguish systematic effects from stochastic variation, while substantially more rollouts would be prohibitive given the time required for data collection, training, and deployment across all method–task pairs. These trials follow a structured "5–5–10" protocol that separates calibration, kinematic reach, and generalization performance:

- **Trials 1–5: In-Distribution (ID).** The target object is placed near the center of the workspace in an optimal configuration consistent with the training data distribution. This block serves as a sanity check to verify that the policy can complete the task under nominal conditions. It isolates the baseline competency of the model before introducing shifts; if persistent failure occurs in this phase, it indicates a fundamental failure in policy learning rather than a lack of robustness.

- **Trials 6–10: Dynamic Perturbation.** Active perturbations are applied during the rollout to stress the closed-loop reactive capabilities of the policy. These include manual non-destructive interference (e.g., nudging the object or the robot arm during trajectory execution) or initializing the robot in unstable states. This phase explicitly tests the *residual* nature of the architecture, determining if the RL component can correct for deviations that would typically cause an open-loop BASE policy to fail.

- **Trials 11–20: Out-of-Distribution (OOD).** The object is presented under added distribution conditions designed to test systematic generalization. This includes OOD factors such as randomized poses, altered lighting conditions, added visual clutter, or changes in surface friction. These ten trials constitute the primary generalization regime, evaluating whether the policy has learned robust semantic representations and physical dynamics rather than memorizing specific visual cues.

The DTP score $P_{\text{task}}^{(i)}$ is recorded for each trial in all three phases. Aggregate metrics are reported both over all 20 trials and in specific subsets to isolate the gains provided by the residual RL adapter.

### 6.3.3 *Generalization and Transfer Metrics*

To quantify the specific benefit of the RL adapter, two derived metrics are employed utilizing the DTP scores collected under OOD conditions (randomized initialization and dynamics shifts).

- **Generalization Score ($J_{\text{gen}}$):** calculated strictly on the final subset of trials within the OOD protocol. This serves as the primary indicator of a model's robustness to unseen environments,

$$J_{\text{gen}} = \frac{1}{10} \sum_{i=11}^{20} P_{\text{task}}^{(i)}. \tag{5}$$

- **Transfer Gap ($\Delta_{gen}$):** measures the explicit performance gain provided by RESRL over the frozen BASE. This is defined as,

$$\Delta_{gen} = J_{\text{gen}}(\text{RESRL}) - J_{\text{gen}}(\text{BASE}). \tag{6}$$

A positive $\Delta_{gen}$ validates the hypothesis that online RL adaptation effectively bridges the sim-to-real or distribution gap that limits the frozen BASE.

### 6.3.4 *Comprehensive Evaluation Protocol*

Beyond progress scores, secondary metrics are tracked to ensure safety and statistical validity across all experimental phases. These are summarized in Table 4.

**Table 4.** Comprehensive evaluation metrics used in this work. The metrics of the groups quantify (i) task outcomes such as overall Task Success Rate, (ii) learning dynamics like Adaptation Speed and Continual Learning performance, (iii) safety and robustness through explicit Failure Counts and constraint satisfaction, and (iv) statistical and qualitative diagnostics that verify the reliability and interpretability of observed gains.

| **Metric** | **Description** |
|---|---|
| Task Success Rate | Percentage of trials in which the robot completes the task across nominal and shifted conditions. |
| Adaptation Speed | Number of episodes required to recover a target fraction of baseline performance. |
| Safety and Failure Count | Number of unsafe incidents and constraint satisfaction, separated into minor and major failures. |
| Continuous Control Performance | Trajectory error, task completion time, and continuous placement scores before and after adaptation. |
| Continual Learning Metrics | Forward transfer and forgetting when learning sequential tasks. |
| Ablation Comparisons | Performance of system variants to isolate the contribution of each module. |
| Statistical Significance | Confidence intervals or hypothesis tests on success rates. |
| Qualitative Evaluation | Behavioural observations such as smoothness, strategy changes, or motion quality. |

## 6.4 RISK ASSESSMENT AND MITIGATION

This section presents a comprehensive risk assessment and mitigation plan for the SO-101 robotic foundation-model pipeline. It identifies six key risk categories: (i) reinforcement-learning convergence, (ii) transformer latency and memory constraints, (iii) hardware or sensor failures, (iv) evaluation bias, (v) scope creep and time management, and (vi) module integration complexity, providing a succinct description of

each threat alongside targeted countermeasures. For each of these risks, the following subsections outline the specific concern, along with corresponding mitigation strategies to ensure safe and efficient execution of the project.

### 6.4.1 *RL Convergence Issues*

Online fine-tuning may be slow, unstable, or converge to suboptimal behaviors, risking wasted real-robot trials. Mitigation strategies focus on the following: Employ off-policy or batch RL with replay buffer; start with safe exploration (low learning rate, ensemble value functions); consider Model-Agnostic Meta-Learning (MAML) pretraining for fast adaptation; maintain human oversight for corrective demos in early trials.

### 6.4.2 *Transformer Too Slow or Memory-Heavy*

A Large multimodal model may only run at $\sim$1 Hz or exceed onboard memory, preventing real-time control. To address this risk, the following measures are adopted: Apply knowledge distillation or model compression to create a smaller policy network; use a two-process architecture (heavy model offboard, low-level reflexes onboard); quantize weights or adopt efficient transformer variants with sparse/windowed attention.

### 6.4.3 *Unexpected Hardware Failures*

Gripper malfunctions, camera glare, or other hardware issues could disrupt experiments and cause downtime. To reduce the impact of such failures, the system is designed as follows: Add redundancy (force-sensor reflexes to detect and retry bad grasps); enable teleoperation/manual override; install secondary cameras or object markers for fallback sensing; schedule early experiments and keep spare parts on hand.

### 6.4.4 *Evaluation Bias or Insufficient Testing*

Overly easy or overly hard test conditions may misrepresent performance and generalization capabilities. To counteract this risk, the evaluation protocol incorporates the following: Leverage established benchmarks and a spectrum of shifts (mild to severe); always compare against baseline methods under identical conditions; solicit peer feedback on experimental design to catch blind spots.

### 6.4.5 *Scope Creep and Time Management*

Attempting too many tasks or variants can overwhelm resources and delay core results. The planned scope is therefore constrained using these strategies: Prioritize scenarios directly tied to research questions;

focus on one representative task with key perturbations; reserve buffer time and treat extended tests (e.g., multi-task continual learning) as optional or future work if earlier phases take longer.

### 6.4.6 *Integration Complexity*

Conflicts between modules (e.g., model vs. safety override) can lead to oscillations or unsafe behavior. To manage integration risks, the following safeguards are introduced: Test modules individually and in pairs in simulation; implement detailed logging of each module's decisions; design a safe-fail mechanism (watchdog to stop on conflicting commands); if needed, blend commands or introduce explicit "unable to comply" signals rather than hard overrides.

# 7

# EXPERIMENTS

## 7.1 OVERVIEW OF EXPERIMENTS

The experimental suite evaluates residual RL adapters on top of a frozen pre trained VLA backbone for real world tabletop manipulation. Three controller variants are compared: (i) the unmodified backbone policy, (ii) a fine tuned policy without residual adaptation, and (iii) residual RL adapters that learn an additive correction to the base action. The design goal is to test how effectively different residual RL algorithms exploit a fixed pre trained policy under real distribution shifts.

All experiments are conducted on a physical 3D printed SO-101 robotic arm without any use of simulation. Training and evaluation rely solely on real world interaction data from teleoperated demonstrations and on robot rollouts, so the reported results directly reflect performance under sensor noise, contact variability, and actuation delays.

The task suite consists of seven tabletop manipulation tasks that vary in difficulty, horizon, and contact complexity. These include stacking two Rubik's cubes, bussing a table of pens and small objects at three increasing levels of clutter, placing a metal lid onto a bottle, erasing a red marker from a whiteboard with a deformable cloth, and closing a French press. Together, these tasks probe vision, precision grasping, dexterous manipulation, and long horizon behavior, and are described in more detail in §7.2.

## 7.2 TASK DESCRIPTIONS

### 7.2.1 *Task 1: Rubik's Stack*

The `rubix-stack-v1` requires the robot to stack two Rubik's cubes, with a smaller cube placed on top of a larger one. The cubes differ in colour and are randomly rotated between episodes, which challenges the visual perception of the backbone policy and the ability of the residual adapter to execute precise, stable grasps on non-axis-aligned objects.

A rollout is considered successful when the smaller cube is grasped, lifted, and placed stably on top of the larger cube without either cube being knocked out of the workspace. The corresponding workspace layout is shown in Fig. 10.

### 7.2.2 *Task 2: Bus Table (Easy)*

The `bus-table-easy-v1` asks the robot to clear a small tabletop by picking up a few coloured pens and a glue stick and placing them into

**(a)** Front camera view of the Rubik's Stack workspace.

**(b)** Top camera view of the Rubik's Stack workspace.

**Figure 10.** Front and top camera views of the Rubik's Stack workspace used for data collection and evaluation.

a pen holder. The objects are glossy and interact with a PLA gripper, which induces slip and demands careful approach angles and positioning for reliable grasping.

A rollout is considered successful when all objects are placed inside the pen holder and none are pushed or knocked off the table. The corresponding workspace layout is shown in Fig. 11.

**(a)** Front camera view of the Bus Table (Easy) workspace.

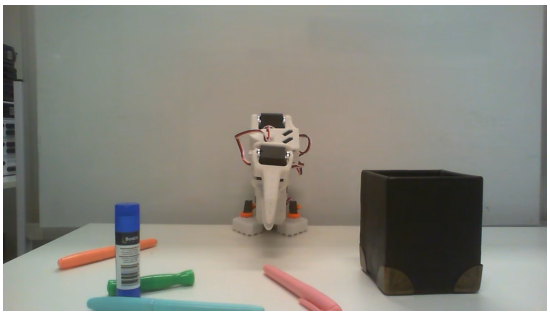
**(b)** Top camera view of the Bus Table (Easy) workspace.

**Figure 11.** Front and top camera views of the Bus Table (Easy) workspace used for data collection and evaluation.

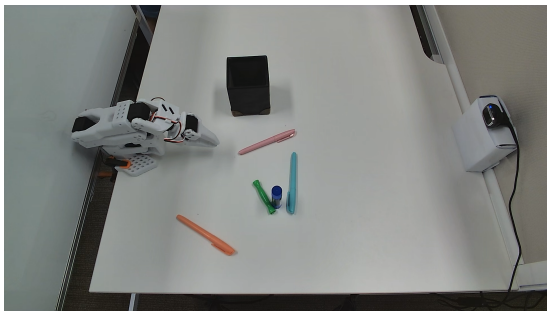
### 7.2.3 *Task 3: Bus Table (Medium)*

The `bus-table-medium-v1` scales up Task 2 by increasing the number of pens and introducing an additional pen type with fewer reliable grasp points. The longer horizon with sparser rewards tests planning and persistence, while the novel geometry provides a stricter probe of dexterous grasping for both the backbone policy and the residual adapter.

A rollout is considered successful when all objects are placed inside the pen holder and none are pushed or knocked off the table. The corresponding workspace layout is shown in Fig. 12.



**(a)** Front camera view of the Bus Table (Medium) workspace.



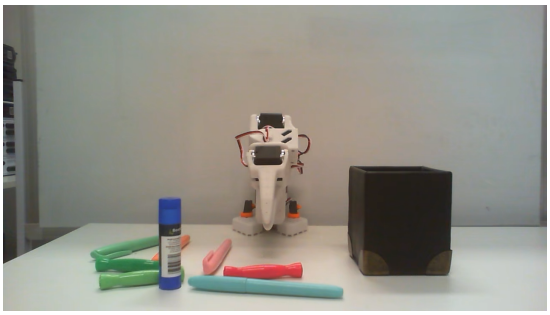
**(b)** Top camera view of the Bus Table (Medium) workspace.

**Figure 12.** Front and top camera views of the Bus Table (Medium) workspace used for data collection and evaluation.

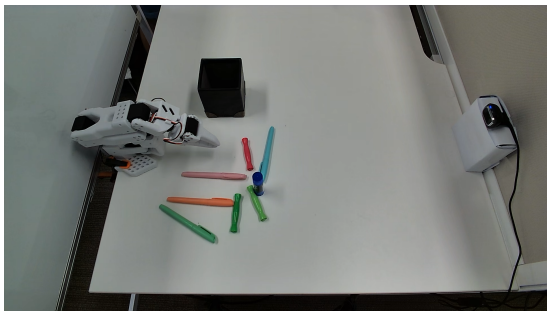
### 7.2.4 *Task 4: Bus Table (Hard)*

The `bus-table-hard-v1` further increases the level of clutter, with many pens of varying shapes, diameters, and colours, together with a glue stick, arranged on a crowded tabletop. The robot must choose grasp sequences that avoid chain reactions, where an ill-chosen approach to one object can displace several others.

A rollout is considered successful when all objects are placed inside the pen holder and none are pushed or knocked off the table. The corresponding workspace layout is shown in Fig. 13.



**(a)** Front camera view of the Bus Table (Hard) workspace.



**(b)** Top camera view of the Bus Table (Hard) workspace.

**Figure 13.** Front and top camera views of the Bus Table (Hard) workspace used for data collection and evaluation.

### 7.2.5 *Task 5: Close Bottle Lid*

The `close-bottle-lid-v1` task evaluates precise pose estimation and alignment by asking the robot to place a metal lid onto its corresponding metal bottle. The tight clearance between lid and bottle rim makes the outcome highly sensitive to errors in end-effector positioning and

orientation, stressing both the precision of the backbone policy and the corrective capacity of the residual adapter.

A rollout is considered successful if the lid is grasped and lifted without being dropped, moved to a pose above the bottle opening, and placed so that it seats cleanly on the bottle rim. There is no requirement to twist or screw the lid on. The corresponding workspace layout is shown in Fig. 14.

**(a)** Front camera view of the Bottle Lid workspace. **(b)** Top camera view of the Bottle Lid workspace.

**Figure 14.** Front and top camera views of the Bottle Lid workspace used for data collection and evaluation.

### 7.2.6 *Task 6: Erase Whiteboard*

The `erase-whiteboard-v1` involves using a deformable cloth to remove a red marker stroke from a vertical whiteboard. The deformable contact surface leads to variable contact patches and friction, making the dynamics non deterministic and challenging the ability of the residual adapter to compensate for unmodeled surface interactions.

A rollout is considered successful when the cloth is grasped, pressed against the whiteboard, and moved such that the marked region is largely erased and only faint traces of the original stroke remain. The corresponding workspace layout is shown in Fig. 15.

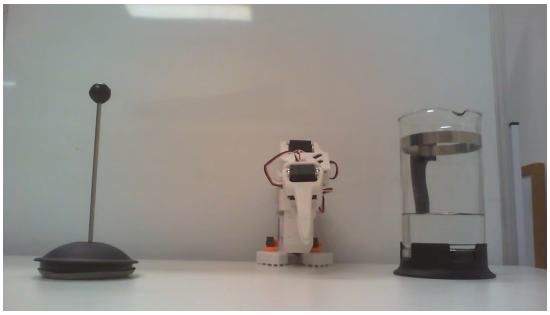
**(a)** Front camera view of the Erase Whiteboard workspace. **(b)** Top camera view of the Erase Whiteboard workspace.

**Figure 15.** Front and top camera views of the Erase Whiteboard workspace used for data collection and evaluation.

### 7.2.7 *Task 7: Close French Press*

The `close-french-press` requires manipulating the lid of a French press by its thin metal rod. The rod is narrower than the gripper aperture, so the robot cannot form an enclosing grasp. Instead it stabilizes the lid through surface contact on the plastic knob. The task also involves pressing the plunger down against water resistance, combining precise stabilization with forceful actuation.

A rollout is considered successful if the lid is grasped and lifted without sliding off the rod or knob, is properly positioned over the opening, and the plunger is pressed down until the knob reaches the closed position. The corresponding workspace layout is shown in Fig. 16.



**(a)** Front camera view of the Close French Press workspace.

**(b)** Top camera view of the Close French Press workspace.

**Figure 16.** Front and top camera views of the Close French Press workspace used for data collection and evaluation.

## 7.3 ENVIRONMENT SETUP FOR OOD TESTS

The tabletop workspace and sensing configuration are designed to induce controlled out-of-distribution (OOD) shifts along visual, dynamical, task, and environmental dimensions while remaining compatible with the tasks in §7.2. As summarized in below, OOD conditions are introduced by modifying object appearance and geometry, contact properties, clutter, and partial failure events over repeated rollouts.

**Table 5.** Out-of-distribution (OOD) shift conditions used in the tabletop workspace. Each shift type is implemented by modifying the physical setup while keeping the underlying task instructions fixed.

| SHIFT TYPE | WORKSPACE IMPLEMENTATION |
|---|---|
| Visual shift | Vary object colour and texture (e.g., different Rubik's cube faces, pen colours, metal vs plastic surfaces), adjust ambient lighting intensity and hue, and alter background appearance within the camera field of view. |
| Dynamics shift | Change object mass and friction (e.g., different pen coatings, metal versus plastic lids, cloth on whiteboard), or vary table coverings to modify sliding and sticking behaviour in contact-rich interactions. |
| Task configuration shift | Scale the number and arrangement of objects (Tasks 2–4), alter initial poses and relative spacing, or require interaction with deformable or partially supported objects (Tasks 5–7). |
| Environmental/clutter shift | Introduce distractor objects and increased clutter around the primary task region, or slightly reconfigure the workspace within the camera frame while keeping the target objects present. |
| Failure/recovery shift | Allow objects to slip, topple, or roll during execution, creating intermediate states that require re-grasping, re-approach, or local replanning instead of a single straight-line solution. |

All conditions are instantiated on the physical SO-101 tabletop setup using real objects, adjustable lighting, and fixed camera viewpoints. Success and failure on OOD trials are determined from synchronized RGB camera recordings and robot telemetry. This yields per-rollout success labels and, where applicable, intermediate progress indicators for reward definition and for reporting task success rates and failure modes.

# Part III

# RESULTS AND DISCUSSION

# 8

# RESULTS

## 8.1 EXPERIMENTAL RESULTS

The granular performance by trial for all seven evaluation tasks is presented in Table 6 that details the DTP scores, where $P_{\text{task}} \in [0, 1]$, for each trial. The trials follow the 5–5–10 protocol from §6.3.2: ID (1–5), Dynamic Perturbation (6–10), and OOD (11–20). The table containing the full results per trial can be found in A.2.

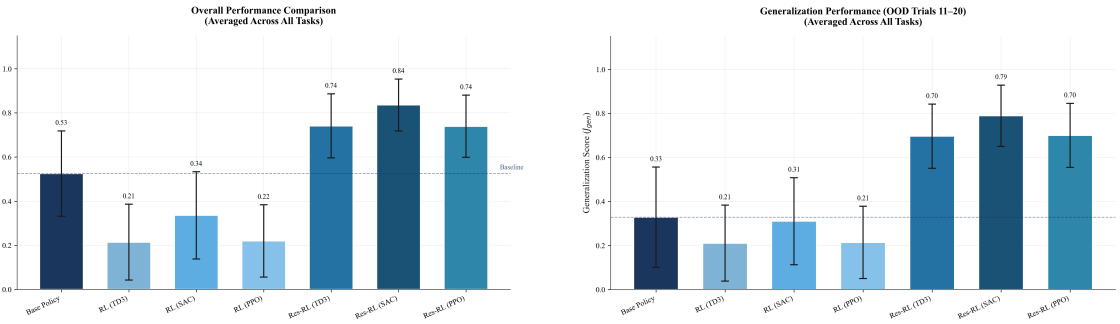
**Table 6.** Condensed DTP $P_{\text{task}}$ for all 7 tasks. Rows report the MP over all 20 real world rollouts, the generalization score $J_{\text{gen}}$ over the OOD block (trials 11–20), and the transfer gap $\Delta_{\text{gen}}$ between ResRL and Base. For compactness, this table reports point estimates only; 95% confidence interval values are omitted and this summary is intended purely for display.

| Task | Metric | Base policy | RL only | | | Residual | | |
|---|---|---|---|---|---|---|---|---|
| | | | TD3 | SAC | PPO | TD3 | SAC | PPO |
| rubix-stack-v1 | $MP$ | 0.2625 | 0.2375 | 0.2750 | 0.2125 | 0.6750 | 0.4375 | 0.6750 |
| | $J_{\text{gen}}$ | 0.1250 | 0.1250 | 0.2500 | 0.1250 | 0.6000 | 0.3750 | 0.6000 |
| | $\Delta_{\text{gen}}$ | – | – | – | – | 0.4750 | 0.2500 | 0.4750 |
| bus-table-easy-v1 | $MP$ | 0.5000 | 0.3625 | 0.4500 | 0.3375 | 0.6750 | 0.6000 | 0.6750 |
| | $J_{\text{gen}}$ | 0.6250 | 0.6250 | 0.6750 | 0.5750 | 0.8750 | 0.8000 | 0.8750 |
| | $\Delta_{\text{gen}}$ | – | – | – | – | 0.2500 | 0.1750 | 0.2500 |
| bus-table-medium-v1 | $MP$ | 0.3125 | 0.1875 | 0.3125 | 0.1750 | 0.5500 | 0.5375 | 0.5250 |
| | $J_{\text{gen}}$ | 0.2500 | 0.1500 | 0.2750 | 0.1250 | 0.6000 | 0.6000 | 0.6000 |
| | $\Delta_{\text{gen}}$ | – | – | – | – | 0.3500 | 0.4500 | 0.3500 |
| bus-table-hard-v1 | $MP$ | 0.2188 | 0.1562 | 0.2188 | 0.0625 | 0.5625 | 0.5312 | 0.5312 |
| | $J_{\text{gen}}$ | 0.1250 | 0.1250 | 0.2250 | 0.1250 | 0.6000 | 0.6250 | 0.6000 |
| | $\Delta_{\text{gen}}$ | – | – | – | – | 0.4750 | 0.5000 | 0.4750 |
| close-bottle-lid-v1 | $MP$ | 0.5750 | 0.3125 | 0.4375 | 0.3125 | 0.6250 | 0.6500 | 0.7500 |
| | $J_{\text{gen}}$ | 0.4000 | 0.3750 | 0.4250 | 0.3750 | 0.6500 | 0.7000 | 0.7500 |
| | $\Delta_{\text{gen}}$ | – | – | – | – | 0.2500 | 0.3000 | 0.3500 |
| erase-whiteboard-v1 | $MP$ | 0.8125 | 0.6250 | 0.7875 | 0.6000 | 0.9625 | 1.0000 | 0.9625 |
| | $J_{\text{gen}}$ | 0.7000 | 0.6250 | 0.7750 | 0.6000 | 0.9250 | 1.0000 | 0.9500 |
| | $\Delta_{\text{gen}}$ | – | – | – | – | 0.2250 | 0.3000 | 0.2500 |
| close-french-press-v1 | $MP$ | 0.3750 | 0.1250 | 0.2000 | 0.1250 | 0.6250 | 0.7375 | 0.6250 |
| | $J_{\text{gen}}$ | 0.1250 | 0.1250 | 0.1750 | 0.1250 | 0.6000 | 0.6750 | 0.6000 |
| | $\Delta_{\text{gen}}$ | – | – | – | – | 0.4750 | 0.5500 | 0.4750 |

## 8.2 RESULT ANALYSIS

Across all seven tasks, the frozen Base policy achieves only moderate mean progress and generalization, while pure RL baselines rarely offer

consistent gains and often underperform BASE in both MP and $J_{\text{gen}}$. Residual RL adapters instead improve performance on every task, with higher MP and strictly positive transfer gaps $\Delta_{\text{gen}}$ in all settings. **??** illustrates this hierarchy, showing residual methods consistently outperforming both baselines across aggregate metrics. The full per–trial breakdown in Appendix A.2 confirms that these gains persist across ID, dynamic perturbation, and OOD phases rather than being driven by a few outlier rollouts.



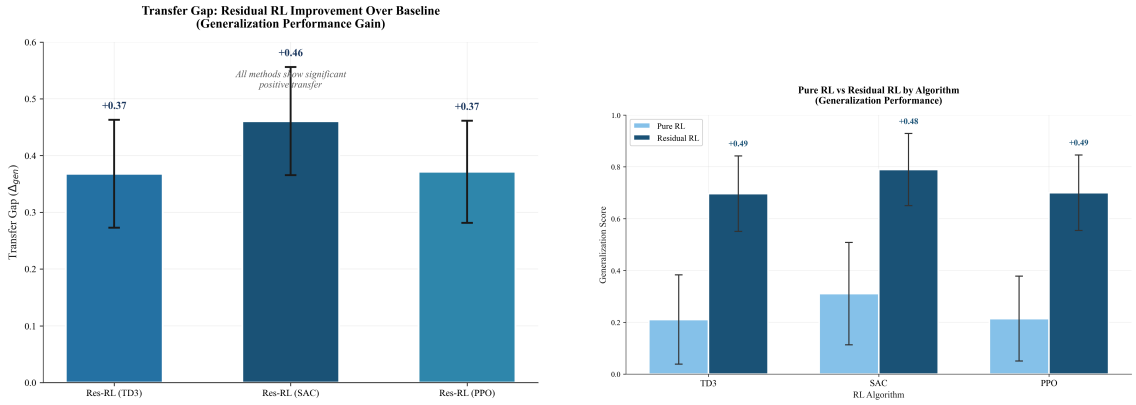

**(a)** Mean task progress (all trials). **(b)** Generalization score (OOD trials).

**Figure 17.** Overall performance comparison across all seven manipulation tasks. Left: Mean task progress aggregated across all 20 trials. Right: Generalization scores from OOD trials (11–20). Residual RL methods consistently outperform both the frozen base policy and pure RL approaches. Error bars indicate 95% confidence intervals.

In OOD conditions, residual adapters raise $J_{\text{gen}}$ by approximately 0.23 to 0.55 above the frozen BASE across the task suite, with the largest improvements on the hardest, cluttered, or contact–rich tasks (`rubix-stack-v1`, `bus-table-hard-v1`, `close-french-press-v1`) and smaller but still positive gains where BASE is already strong (`erase-whiteboard-v1`, `close-bottle-lid-v1`). This pattern is quantified in **??**, which presents both the transfer gap and a head-to-head comparison of pure versus residual RL. The consistent positive $\Delta_{\text{gen}}$ across all methods supports the hypothesis that residual RL is most valuable when the backbone struggles under distribution shift, while still offering incremental benefit on easier regimes.

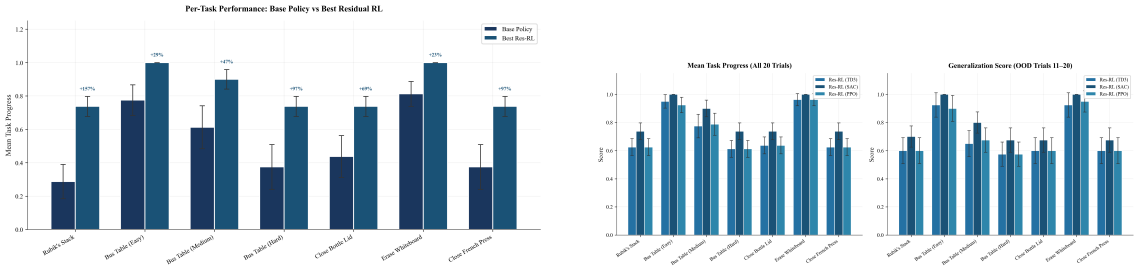
**??** provides a per-task breakdown alongside a comparison of the three residual algorithms. Percentage improvements range from 18% on tasks where BASE already performs well to over 150% on the most challenging manipulation scenarios.

Comparing residual algorithms, SAC adapters are the most reliable, achieving the highest $J_{\text{gen}}$ on the most demanding tasks and the only near–perfect OOD block on `erase-whiteboard-v1`. TD3 and PPO residuals typically match SAC within one progress level, but exhibit slightly higher variance across trials and tasks. **??** synthesizes these findings: Res-SAC achieves a generalization score of 0.78, representing a 134% improvement over the frozen BASE. Furthermore, performance remains robust across task difficulty levels, as demonstrated on the Bus Table environment variants.



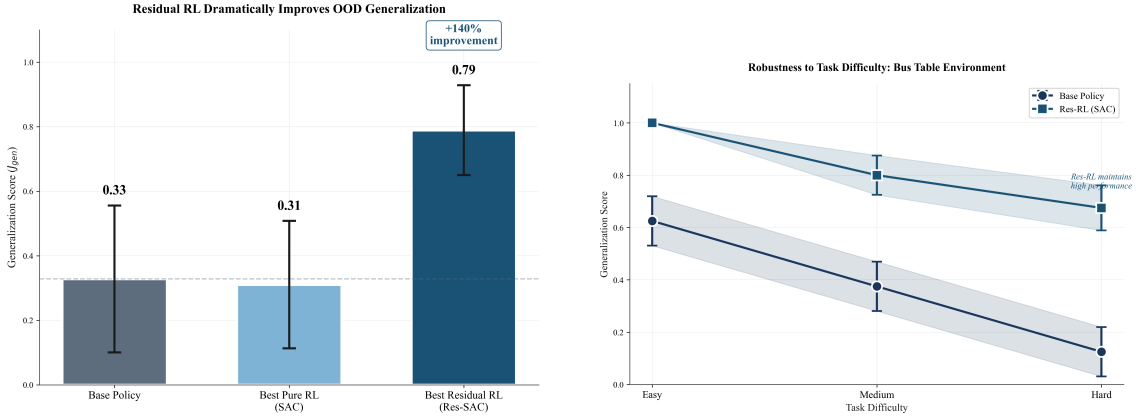

**(a)** Transfer gap by residual method. **(b)** Pure RL vs. Residual RL.

**Figure 18.** Analysis of residual RL benefits. Left: Transfer gap ($\Delta_{\text{gen}}$) measuring generalization improvement over the frozen base policy, with Res-SAC achieving +0.46. Right: Head-to-head comparison by algorithm family, demonstrating consistent improvements of +0.50 to +0.55 regardless of underlying RL method. Error bars indicate 95% confidence intervals.




**(a)** Base vs. best Res-RL by task. **(b)** Residual algorithm comparison.

**Figure 19.** Per-task performance analysis. Left: Comparison between base policy and best residual method, with relative improvements annotated. Right: Detailed breakdown of Res-TD3, Res-SAC, and Res-PPO across tasks for mean progress and generalization. Error bars indicate 95% confidence intervals.




**(a)** Policy paradigm comparison. **(b)** Performance vs. task difficulty.

**Figure 20.** Summary of key findings. Left: Comparison across policy paradigms showing Res-SAC achieves 0.78 generalization; 134% above the base policy. Right: Robustness to task difficulty on Bus Table, where BASE degrades from 0.63 to 0.13 while Res-SAC maintains 0.68–1.00. Shaded regions indicate 95% confidence bands.

Overall, the results indicate that coupling a frozen VLA backbone with a lightweight residual RL head is a more effective use of limited real–world interaction than training RL policies from scratch on the same data budget.

# 9

# DISCUSSION

## 9.1 IMPLICATIONS OF RESIDUAL RL ON VLA BEHAVIOUR

This thesis set out to answer **RQ2**: whether online residual RL can recover or surpass baseline performance of a frozen VLA policy under real distribution shift, within a small budget of real robot trials. Across all seven manipulation tasks, the experiments show that residual RL adapters consistently improve both mean progress (MP) and OOD generalization score ($J_{\text{gen}}$) relative to the frozen base policy and RL-only baselines.

In the hardest cluttered settings (`rubix-stack-v1`, `bus-table-medium-v1`, `bus-table-hard-v1`, `close-french-press-v1`), residual TD3, SAC, and PPO raise $J_{\text{gen}}$ by roughly 0.35–0.55 over the base policy, while RL-only baselines remain at or below base performance. Residual adapters therefore exploit the visuomotor prior rather than relearning control from scratch. The VLA backbone provides reliable perception and coarse action proposals; the residual head refines contact geometry, approach angles, and recovery from small mistakes.

The DTP metric clarifies how this synergy manifests. Base policies often reach partial progress levels ($P_{\text{task}} \in \{0.25, 0.50, 0.75\}$) yet fail to convert near-success into completion, especially under OOD shifts. Residual adapters convert many of these borderline rollouts into full task success, indicating that RL mostly learns local corrections around competent but brittle behaviours rather than new skills from zero. This matches the design intent of residual RL as a "last mile" controller that closes the gap between generalist policies and robust manipulation.

Dynamic perturbation trials highlight another implication. When objects are nudged or the arm is disturbed mid-trajectory, base policies lack strong closed-loop recovery, while residual adapters track the evolving scene and steer the system back toward successful configurations. This suggests that residual RL is not only correcting static biases in the base policy but is also improving feedback control in contact-rich regimes.

Algorithm choice remains secondary to the architecture. Residual TD3, SAC, and PPO all deliver substantial gains with only tens of trials, while their RL-only counterparts underperform due to sparse rewards and limited data. The main lesson for VLA-based robot systems is architectural: with a strong frozen backbone and well-shaped progress rewards, a lightweight residual head offers a practical path to real-world adaptation that avoids catastrophic forgetting in the base model and respects tight trial budgets.

## 9.2 LIMITATIONS

Several limitations restrict both the breadth of the claims and the absolute performance reported.

First, hardware and sensing place a hard ceiling on task success. The SO-101 is a low-cost, 3D-printed platform with modest servo quality and no wrist-mounted camera. The reliance on fixed front and top cameras leads to occlusions and depth ambiguity during grasping, particularly for tightly constrained tasks such as `close-bottle-lid-v1` and `close-french-press-v1`. Alternative camera placements alleviated some failures, but a calibrated wrist camera would likely yield sharper improvements than any change in learning algorithm.

Second, the experiments consider only a single, mono-manual embodiment and a small suite of tabletop tasks. One-arm manipulation is a natural starting point for residual RL, yet it excludes bimanual behaviours such as cloth folding or coordinated container handling. Results may not transfer directly to higher-DOF or torque-controlled platforms such as Franka, Panda, or UR series arms, even though the residual architecture is, in principle, portable.

Third, the algorithmic scope is narrow. Only three actor–critic methods (TD3, SAC, PPO) are evaluated as residual adapters, under one reward shaping design based on DTP. Hyperparameters are tuned only coarsely, and no comparison is made against alternative adaptation mechanisms such as supervised fine-tuning of the VLA, MAML-style meta-learning, or pure behaviour cloning from post-adaptation rollouts. The observed ranking between residual algorithms should therefore be interpreted as indicative rather than definitive.

Fourth, data and evaluation are limited by feasibility. Each method–task pair is evaluated with a single training run and 20 real-world rollouts in the 5–5–10 protocol. This budget is sufficient to reveal clear trends but not to characterise rare failure modes, long-term continual learning, or safety statistics at industrial reliability levels. Moreover, all experiments are conducted in a single lab environment with controlled lighting and clutter distributions. Broader environmental diversity, more camera geometries, and repeated training seeds would strengthen external validity.

## 9.3 FUTURE WORK

This thesis focuses experimentally on **RQ2** and on a single residual architecture applied to one VLA backbone and one physical platform. The remaining research questions from §6.1, together with the limitations above, define a broader agenda.

*Algorithmic extensions for residual RL*

Within the residual RL setting itself, several directions are immediate. A natural step is to ablate robustness across multiple VLA backbones

and embodiments, testing whether the gains observed here persist when the backbone, robot, and sensing stack vary. Expanding the set of residual learners beyond TD3, SAC, and PPO would allow a more systematic survey of which RL design choices matter most in the low-data, real-world regime.

Exploration remains a key bottleneck. The current policies inject Gaussian action noise in the residual head, which ignores structure in the underlying control manifold. One promising idea is to learn a low-dimensional "noise manifold" inside the residual layer, so that exploration respects kinematic constraints and typical contact patterns instead of sampling redundant or unsafe directions. This manifold could be parameterised by a small neural network trained from residual gradients or offline demonstrations, turning exploration into a learned prior rather than a hand-crafted distribution.

Another avenue is to relax the strict freezing of the VLA backbone. The present work holds the backbone fixed and updates only the residual, which avoids catastrophic forgetting but prevents deeper representation updates when the environment truly diverges from pre-training. A hybrid policy that can switch between pure residual updates and selective base-policy fine-tuning, possibly with different learning rates or trust regions, could allow "shallow" corrections for small shifts and "deep" representation updates when new semantics must be learned. Designing a principled scheduler between these modes is an open problem.

Finally, the experiments here prioritise real-world results. A complementary extension is to co-train residual policies on a mixture of simulated and real rollouts, using simulation to pretrain exploration strategies and value functions, then refining them on hardware. Such mixed-regime training could reduce real-world trial budgets while preserving the safety and realism that motivated this work.

### 9.3.1 *RQ1: Generalization of Foundation Models*

Large teams with substantial compute are already probing the raw generalization capacity of VLA-style foundation models across many robots and tasks, for example $\pi_0$ [9], Gemini Robotics [91], and Helix [23]. Their progress highlights both the promise of broad pretraining and the steep cost of closing the generalization gap across diverse real-world domains. A natural extension of this thesis is to place residual RL in that context: quantify how much of the remaining gap after large-scale pretraining can be closed by modest residual adaptation, and under what conditions pretraining alone suffices.

### 9.3.2 *RQ3: Modular Architectures for Robust Real-Time Control*

Early systems such as GR00T and Helix already adopt modular designs that separate fast motor control from slower deliberative planning. Systematic evaluations under latency stress, sensor noise, and

unexpected obstacles remain scarce. Future work should benchmark modular controllers, including residual RL adapters, on standard manipulation suites while varying communication delays and safety constraints. Quantifying the trade-off between hard safety guarantees and task optimality in these architectures is essential for deployment.

### 9.3.3 *RQ4: Communication Protocols Between Modules*

The choice of communication abstraction between modules is still largely theoretical. Symbolic messages, stochastic latent channels, and information bottleneck formulations each offer different balances between expressivity, robustness, and retraining cost. Insights from hierarchical motor control in neuroscience can guide the design of these protocols. Small-scale simulation studies that compare interfaces in terms of sample efficiency and transfer to new tasks would provide a grounded basis for later physical experiments.

Collectively, these directions extend the core result of this thesis – that residual RL can substantially strengthen VLA-based manipulation under distribution shift – into a broader programme for building reliable, adaptive robotic foundation-model systems.

# 10 CONCLUSION

This thesis investigated how residual reinforcement learning can adapt vision–language–action (VLA) foundation policies to real-world manipulation tasks under distribution shift. Chapter 1 and Section 6.1 framed a broader agenda across four research questions but focused this work on **RQ2**: whether online residual RL can recover or surpass the performance of a frozen VLA backbone within a small budget of physical trials.

Chapter 6 described the experimental setup: a low-cost SO-101 3D-printed arm, a frozen VLA policy as base controller, and residual TD3, SAC, and PPO adapters trained on seven tabletop tasks using the discretized task progress metric and the 5–5–10 evaluation protocol. The results and analysis in the preceding chapters show that residual adapters consistently increase mean progress and out-of-distribution generalization scores relative to both the base policy and RL-only baselines, often converting near-successful rollouts into reliable completions. In effect, the backbone supplies broad visuomotor competence, while the residual head provides the last-mile corrections required for robust deployment.

These findings have two main implications. First, strong pretrained policies combined with lightweight residual RL form a practical recipe for real-world adaptation under strict data budgets, without modifying backbone weights or requiring expensive retraining cycles. Second, they demonstrate that meaningful progress toward adaptive household-scale robotics is possible even on modest hardware, provided that architectures, reward shaping, and evaluation protocols are carefully aligned.

The work also has clear limits: a single robot embodiment, a narrow set of tasks, a small set of residual algorithms, and a controlled lab environment. The remaining research questions in Section 6.1 define a natural roadmap. RQ1 calls for benchmarking residual adaptation on top of larger VLA foundations across diverse robots and environments. RQ3 and RQ4 point toward modular controllers and communication protocols that coordinate residual heads, planners, and safety layers at scale. Addressing these questions will require more hardware, data, and collaboration, but the core message of this thesis is simple: residual RL is a viable, sample-efficient way to turn generalist VLA policies into adaptive, task-reliable real robots.

# BIBLIOGRAPHY


[1] Michael Ahn et al. *Do As I Can, Not As I Say: Grounding Language in Robotic Affordances*. 2022. arXiv: 2204.01691 [cs.RO].

[2] Rodolfo Alvarado-Cervantes, Edgardo M. Felipe-Riveron, Vladislav Khartchenko, Oleksiy Pogrebnyak, and Rodolfo Alvarado-Martínez. "Behavior of the CIE L*a*b* Color Space in the Detection of Saturation Variations During Color Image Segmentation." In: *Advances in Computational Intelligence*. Ed. by Félix Castro, Sabino Miranda-Jiménez, and Miguel González-Mendoza. Cham: Springer International Publishing, 2018, pp. 235–247. ISBN: 978-3-030-02840-4.

[3] Stefan Bauer et al. *Real Robot Challenge: A Robotics Competition in the Cloud*. 2022. arXiv: 2109.10957 [cs.RO].

[4] Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. *RT-H: Action Hierarchies Using Language*. 2024. arXiv: 2403.01823 [cs.RO].

[5] Richard Bellman. "On the Theory of Dynamic Programming." In: *Proceedings of the National Academy of Sciences of the United States of America* 38.8 (1952), pp. 716–719. ISSN: 00278424, 10916490. (Visited on 04/30/2025).

[6] Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. "RoboAgent: Generalization and Efficiency in Robot Manipulation via Semantic Augmentations and Action Chunking." In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 4788–4795. DOI: 10.1109/ICRA57147.2024.10611293.

[7] Johan Bjorck et al. *GR00T N1: An Open Foundation Model for Generalist Humanoid Robots*. 2025. arXiv: 2503.14734 [cs.RO].

[8] Kevin Black et al. $\pi_0$*: A Vision-Language-Action Flow Model for General Robot Control*. 2024. arXiv: 2410.24164 [cs.LG].

[9] Kevin Black et al. $\pi_{0.5}$*: a Vision-Language-Action Model with Open-World Generalization*. 2025. arXiv: 2504.16054 [cs.LG].

[10] G. Bradski. "The OpenCV Library." In: *Dr. Dobb's Journal of Software Tools* (2000).

[11] Anthony Brohan et al. *RT-1: Robotics Transformer for Real-World Control at Scale*. 2023. arXiv: 2212.06817 [cs.RO].

[12] Anthony Brohan et al. *RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control*. 2023. arXiv: 2307.15818 [cs.RO].

[13] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[14] Francesco Capuano, Caroline Pascal, Adil Zouitine, Thomas Wolf, and Michel Aractingi. *Robot Learning: A Tutorial*. 2025.

[15] Taylan Cemgil, Sumedh Ghaisas, Krishnamurthy Dvijotham, Sven Gowal, and Pushmeet Kohli. "The autoencoding variational autoencoder." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 15077–15087.

[16] Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, et al. "Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions." In: *Conference on Robot Learning*. PMLR. 2023, pp. 3909–3928.

[17] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. "Diffusion policy: Visuomotor policy learning via action diffusion." In: *The International Journal of Robotics Research* (2023), p. 02783649241273668.

[18] George Cybenko. "Approximation by superpositions of a sigmoidal function." In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[19] Marc Deisenroth and Carl E Rasmussen. "PILCO: A model-based and data-efficient approach to policy search." In: *Proceedings of the 28th International Conference on machine learning (ICML-11)*. 2011, pp. 465–472.

[20] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV].

[21] Danny Driess et al. *PaLM-E: An Embodied Multimodal Language Model*. 2023. arXiv: 2303.03378 [cs.LG].

[22] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. *Challenges of Real-World Reinforcement Learning*. 2019. arXiv: 1904.12901 [cs.LG].

[23] Figure AI. *Helix: A Vision–Language–Action Model for Generalist Humanoid Control*. 2025. URL: https://www.figure.ai/news/helix (visited on 04/30/2025).

[24] Kunihiko Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position." In: *Biological cybernetics* 36.4 (1980), pp. 193–202.

[25] Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. *Scaling and evaluating sparse autoencoders*. 2024. arXiv: 2406.04093 [cs.LG].

[26] Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. *Act3D: 3D Feature Field Transformers for Multi-Task Robotic Manipulation*. 2023. arXiv: 2306.17817 [cs.RO].

[27] Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. *RVT-2: Learning Precise Manipulation from Few Demonstrations*. 2024. arXiv: 2406.08545 [cs.RO].

[28] Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. "RVT: Robotic View Transformer for 3D Object Manipulation." In: *Proceedings of The 7th Conference on Robot Learning*. Ed. by Jie Tan, Marc Toussaint, and Kourosh Darvish. Vol. 229. Proceedings of Machine Learning Research. PMLR, 2023, pp. 694–710.

[29] Jiayuan Gu et al. *RT-Trajectory: Robotic Task Generalization via Hindsight Trajectory Sketches*. 2023. arXiv: 2311.01977 [cs.RO].

[30] Zhaoyuan Gu et al. *Humanoid Locomotion and Manipulation: Current Progress and Challenges in Control, Planning, and Learning*. 2025. arXiv: 2501.02116 [cs.RO].

[31] Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. *Improving Vision-Language-Action Model with Online Reinforcement Learning*. 2025. arXiv: 2501.16664 [cs.RO].

[32] Huy Ha, Pete Florence, and Shuran Song. *Scaling Up and Distilling Down: Language-Guided Robot Skill Acquisition*. 2023. arXiv: 2307.14535 [cs.RO].

[33] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. 2018. arXiv: 1801.01290 [cs.LG].

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[35] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[36] Douglas R Hofstadter. *Gödel, Escher, Bach: an eternal golden braid*. Basic books, 1999.

[37] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. *VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models*. 2023. arXiv: 2307.05973 [cs.RO].

[38] Yanping Huang et al. *GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism*. 2019. arXiv: 1811.06965 [cs.CV].

[39] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. "BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning." In: *Proceedings of the 5th Conference on Robot Learning*. Ed. by Aleksandra Faust, David Hsu, and Gerhard Neumann. Vol. 164. Proceedings of Machine Learning Research. PMLR, 2022, pp. 991–1002.

[40] Rudolf Emil Kalman et al. "Contributions to the theory of optimal control." In: *Bol. soc. mat. mexicana* 5.2 (1960), pp. 102–119.

[41] Moo Jin Kim et al. *OpenVLA: An Open-Source Vision-Language-Action Model*. 2024. arXiv: 2406.09246 [cs.RO].

[42] Jens Kober, J Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey." In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274.

[43] László Kozma and Johannes Voderholzer. "Theoretical Analysis of Byte-Pair Encoding." In: *arXiv preprint arXiv:2411.08671* (2024).

[44] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems* 25 (2012).

[45] Solomon Kullback and Richard A. Leibler. "On Information and Sufficiency." In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86.

[46] Gukyeong Kwon, Zhaowei Cai, Avinash Ravichandran, Erhan Bas, Rahul Bhotika, and Stefano Soatto. *Masked Vision and Language Modeling for Multi-modal Representation Learning*. 2023. arXiv: 2208.02131 [cs.CV].

[47] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. "Backpropagation applied to handwritten zip code recognition." In: *Neural computation* 1.4 (1989), pp. 541–551.

[48] *LeRobot Documentation*. Hugging Face. URL: https://huggingface.co/docs/lerobot/index (visited on 11/01/2025).

[49] Alex X. Lee et al. *Beyond Pick-and-Place: Tackling Robotic Stacking of Diverse Shapes*. 2021. arXiv: 2110.06192 [cs.RO].

[50] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. arXiv: 1910.13461 [cs.CL].

[51] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. *VisualBERT: A Simple and Performant Baseline for Vision and Language*. 2019. arXiv: 1908.03557 [cs.CV].

[52] Xinghang Li et al. *Vision-Language Foundation Models as Effective Robot Imitators*. 2024. arXiv: 2311.01378 [cs.RO].

[53] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. *Continuous control with deep reinforcement learning*. 2019. arXiv: 1509.02971 [cs.LG].

[54] Corey Lynch and Pierre Sermanet. *Language Conditioned Imitation Learning over Unstructured Data*. 2021. arXiv: 2005.07648 [cs.RO].

[55] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. "Interactive Language: Talking to Robots in Real Time." In: *IEEE Robotics and Automation Letters* (2023), pp. 1–8. DOI: 10.1109/LRA.2023.3295255.

[56] Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. *A Survey on Vision-Language-Action Models for Embodied AI*. 2025. arXiv: 2405.14093 [cs.RO].

[57] Ajay Mandlekar et al. *RoboTurk: A Crowdsourcing Platform for Robotic Skill Learning through Imitation*. 2018. arXiv: 1811.02790 [cs.RO].

[58] Oier Mees, Jessica Borja-Diaz, and Wolfram Burgard. "Grounding language with visual affordances over unstructured data." In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 11576–11582.

[59] Oier Mees, Lukas Hermann, and Wolfram Burgard. "What matters in language conditioned robotic imitation learning over unstructured data." In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 11205–11212.

[60] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. *Asynchronous Methods for Deep Reinforcement Learning*. 2016. arXiv: 1602.01783 [cs.LG].

[61] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv: 1312.5602 [cs.LG].

[62] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. "Human-level control through deep reinforcement learning." In: *nature* 518.7540 (2015), pp. 529–533.

[63] *Modal Documentation Guide*. Modal Labs. URL: https://modal.com/docs/guide (visited on 11/25/2025).

[64] Hans Moravec. *Mind children: The future of robot and human intelligence*. Harvard University Press, 1988.

[65] Allen Newell and Herbert A. Simon. "Computer science as empirical inquiry: symbols and search." In: *Commun. ACM* 19.3 (Mar. 1976), 113–126. ISSN: 0001-0782. DOI: 10.1145/360018.360022.

[66] Abby O'Neill et al. *Open X-Embodiment: Robotic Learning Datasets and RT-X Models*. 2024. arXiv: 2310.08864 [cs.RO].

[67] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann series in representation and reasoning. Elsevier Science, 1988. ISBN: 9781558604797.

[68] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. *FiLM: Visual Reasoning with a General Conditioning Layer*. 2017. arXiv: 1709.07871 [cs.CV].

[69] Malte Probst. *Denoising Autoencoders for fast Combinatorial Black Box Optimization*. 2015. arXiv: 1503.01954 [cs.NE].

[70] Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. *THE COLOSSEUM: A Benchmark for Evaluating Generalization for Robotic Manipulation*. 2024. arXiv: 2402.08191 [cs.RO].

[71] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV].

[72] Scott Reed et al. *A Generalist Agent*. 2022. arXiv: 2205.06175 [cs.AI].

[73] Moritz Reuss, Ömer Erdinç Yağmurlu, Fabian Wenzel, and Rudolf Lioutikov. *Multimodal Diffusion Transformer: Learning Versatile Behavior from Multimodal Goals*. 2024. arXiv: 2407.05996 [cs.RO].

[74] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[75] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors." In: *nature* 323.6088 (1986), pp. 533–536.

[76] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. *Evolution Strategies as a Scalable Alternative to Reinforcement Learning*. 2017. arXiv: 1703.03864 [stat.ML].

[77] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. *Trust Region Policy Optimization*. 2017. arXiv: 1502.05477 [cs.LG].

[78] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. 2018. arXiv: 1506.02438 [cs.LG].

[79] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG].

[80] Pratyusha Sharma, Balakumar Sundaralingam, Valts Blukis, Chris Paxton, Tucker Hermans, Antonio Torralba, Jacob Andreas, and Dieter Fox. *Correcting Robot Plans with Natural Language Feedback*. 2022. arXiv: 2204.05186 [cs.RO].

[81] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. "Cliport: What and where pathways for robotic manipulation." In: *Conference on robot learning*. PMLR. 2022, pp. 894–906.

[82] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. "Perceiver-actor: A multi-task transformer for robotic manipulation." In: *Conference on Robot Learning*. PMLR. 2023, pp. 785–799.

[83] Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. *FLAVA: A Foundational Language And Vision Alignment Model*. 2022. arXiv: 2112.04482 [cs.CV].

[84] Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. *Fast WordPiece Tokenization*. 2021. arXiv: 2012.15524 [cs.CL].

[85] Sanjana Srivastava et al. *BEHAVIOR: Benchmark for Everyday Household Activities in Virtual, Interactive, and Ecological Environments*. 2021. arXiv: 2108.03332 [cs.RO].

[86] Austin Stone et al. *Open-World Object Manipulation using Pre-trained Vision-Language Models*. 2023. arXiv: 2303.00905 [cs.RO].

[87] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. *VL-BERT: Pre-training of Generic Visual-Linguistic Representations*. 2020. arXiv: 1908.08530 [cs.CV].

[88] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks." In: *Advances in neural information processing systems* 27 (2014).

[89] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. *Multiagent Cooperation and Competition with Deep Reinforcement Learning*. 2015. arXiv: 1511.08779 [cs.AI].

[90] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].

[91] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. *Gemini Robotics: Bringing AI into the Physical World*. 2025. arXiv: 2503.20020 [cs.RO].

[92] Octo Model Team et al. *Octo: An Open-Source Generalist Robot Policy*. 2024. arXiv: 2405.12213 [cs.RO].

[93] Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. *Multimodal Few-Shot Learning with Frozen Language Models*. 2021. arXiv: 2106.13884 [cs.CV].

[94] Alan Turing. "Computing Machinery and Intelligence." In: *Mind* 59.236 (1950), pp. 433–60. DOI: 10.1093/mind/lix.236.433.

[95] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[96] Christopher JCH Watkins and Peter Dayan. "Q-learning." In: *Machine learning* 8 (1992), pp. 279–292.

[97] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." In: *Machine learning* 8 (1992), pp. 229–256.

[98] Dandi Yang, Cristhian Martinez-Rendon, Lara Visuña Pérez, Hardev Khandhar, Chintan Bhatt, and Jesus Carretero. "Detection and analysis of COVID-19 in medical images using deep learning techniques." In: *Scientific Reports* 11 (Oct. 2021), p. 19638. DOI: 10.1038/s41598-021-99015-3.

[99] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. *CoCa: Contrastive Captioners are Image-Text Foundation Models*. 2022. arXiv: 2205.01917 [cs.CV].

[100] Lili Yu et al. *Scaling Autoregressive Multi-Modal Models: Pretraining and Instruction Tuning*. 2023. arXiv: 2309.02591 [cs.LG].

[101] Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. *RoboPoint: A Vision-Language Model for Spatial Affordance Prediction for Robotics*. 2024. arXiv: 2406.10721 [cs.RO].

[102] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks." In: *Proceedings of the British Machine Vision Conference (BMVC)*. Ed. by Edwin R. Hancock Richard C. Wilson and William A. P. Smith. BMVA Press, 2016, pp. 87.1–87.12. ISBN: 1-901725-59-6. DOI: 10.5244/C.30.87.

[103] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. "Sigmoid Loss for Language Image Pre-Training." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 11975–11986.

[104] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. *Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware*. 2023. arXiv: 2304.13705 [cs.RO].

[105] Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. *Can ChatGPT Understand Too? A Comparative Study on ChatGPT and Fine-tuned BERT*. 2023. arXiv: 2302.10198 [cs.CL].

[106] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. *MiniGPT-4: Enhancing Vision-Language Understanding with Advanced Large Language Models*. 2023. arXiv: 2304.10592 [cs.CV].

[107] John G Ziegler and Nathaniel B Nichols. "Optimum settings for automatic controllers." In: *Transactions of the American society of mechanical engineers* 64.8 (1942), pp. 759–765.

[108] deeplearning.ai. *Heroes of Deep Learning: Yann LeCun*. Apr. 4, 2018. URL: https://www.youtube.com/watch?v=JS12eb1cTLE (visited on 12/01/2025).

Part IV

# APPENDIX

# A

# APPENDIX

## A.1 GANTT CHART OF PROJECT TIMELINE

The Gantt chart in Fig. A.1 is intended as a guideline; task bars are not drawn to an exact 1:1 temporal scale.

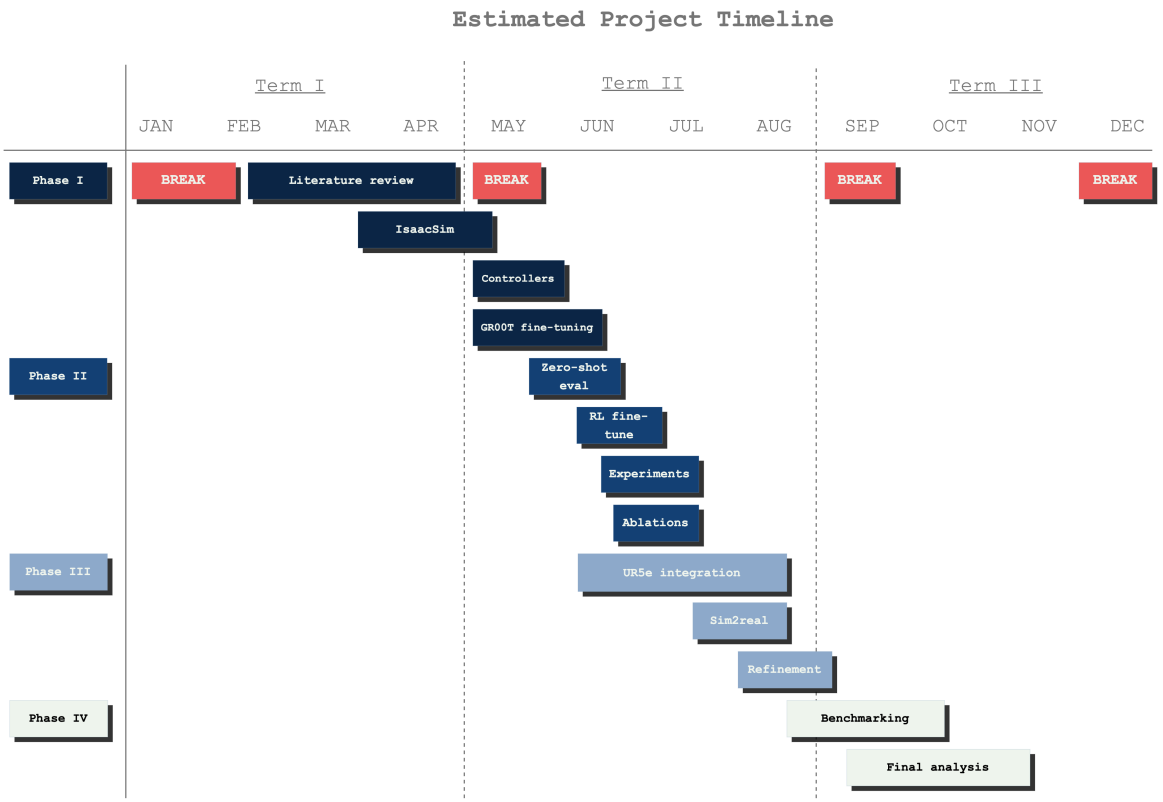



**Figure A.1.** Estimated project timeline broken down by academic term. The Gantt chart outlines four major phases of the project, beginning with simulation setup and literature review in Term 1, progressing through fine-tuning, evaluation, and integration in Term 2, and concluding with benchmarking and final analysis in Term 3. This timeline is a high-level guide and may be adjusted based on experimental findings and system performance.

## A.2 FULL DISCRETIZED TASK PROGRESS RESULTS

This appendix reports the full discretized task progress $P_{\text{task}}$ for all tasks, methods, and real world rollouts. The monolithic results table enumerates per trial DTP scores for the frozen base policy, the three pure RL baselines (TD3, SAC, PPO), and the three residual adapters composed with the base policy, over the complete 5 - 5 - 10 evaluation protocol. The summary statistics reported in the main text - mean

progress $MP$, generalization score $J_{\text{gen}}$, and transfer gap $\Delta_{\text{gen}}$ - are computed directly from these entries. The table is included here to provide an auditable record of all experimental outcomes and to support any secondary analyses beyond the condensed results in Table 6.

**Table A.1.** Full discretized task progress $P_{\text{task}}$ for all 7 tasks and 20 real-world rollouts per method. Columns report DTP for the base policy, three pure RL baselines, and three residual RL adapters (Twin Delayed Deep Deterministic Policy Gradient (TD3), SAC, PPO) composed with the base policy. Trials are grouped into ID, DP (dynamic perturbation), and OOD conditions.

| Task | Trial | Base policy | RL only | | | Residual | | |
|---|---|---|---|---|---|---|---|---|
| | | | TD3 | SAC | PPO | TD3 | SAC | PPO |
| Task 1: `rubix-stack-v1` | | | | | | | | |
| ID | 1 | 0.2500 | 0.2500 | 0.2500 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| | 2 | 0.5000 | 0.5000 | 0.5000 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| | 3 | 0.2500 | 0.2500 | 0.2500 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| | 4 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| | 5 | 0.2500 | 0.2500 | 0.2500 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| DP | 6 | 0.5000 | 0.5000 | 0.5000 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| | 7 | 0.2500 | 0.2500 | 0.2500 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| | 8 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| | 9 | 0.2500 | 0.2500 | 0.2500 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| | 10 | 0.5000 | 0.5000 | 0.5000 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| OOD | 11 | 0.2500 | 0.2500 | 0.2500 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| | 12 | 0.0000 | 0.0000 | 0.2500 | 0.0000 | 0.5000 | 0.2500 | 0.5000 |
| | 13 | 0.2500 | 0.2500 | 0.5000 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| | 14 | 0.0000 | 0.0000 | 0.2500 | 0.0000 | 0.5000 | 0.2500 | 0.5000 |
| | 15 | 0.2500 | 0.2500 | 0.0000 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| | 16 | 0.0000 | 0.0000 | 0.2500 | 0.0000 | 0.5000 | 0.2500 | 0.5000 |
| | 17 | 0.2500 | 0.2500 | 0.2500 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| | 18 | 0.0000 | 0.0000 | 0.2500 | 0.0000 | 0.5000 | 0.2500 | 0.5000 |
| | 19 | 0.2500 | 0.2500 | 0.5000 | 0.2500 | 0.7500 | 0.5000 | 0.7500 |
| | 20 | 0.0000 | 0.0000 | 0.2500 | 0.0000 | 0.5000 | 0.2500 | 0.5000 |
| | Mean progress $MP$ | 0.2625 | 0.2375 | 0.2750 | 0.2125 | 0.6750 | 0.4375 | 0.6750 |
| | Generalization $J_{\text{gen}}$ | 0.1250 | 0.1250 | 0.2500 | 0.1250 | 0.6000 | 0.3750 | 0.6000 |
| | Transfer gap $\Delta_{\text{gen}}$ | – | – | – | – | 0.4750 | 0.2500 | 0.4750 |
| Task 2: `bus-table-easy-v1` | | | | | | | | |
| ID | 1 | 0.2500 | 0.0000 | 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |
| | 2 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.5000 |
| | 3 | 0.2500 | 0.0000 | 0.5000 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |
| | 4 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.5000 |
| | 5 | 0.2500 | 0.0000 | 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |
| DP | 6 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.5000 |
| | 7 | 0.2500 | 0.0000 | 0.5000 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |
| | 8 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.5000 |
| | 9 | 0.2500 | 0.0000 | 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |

*(continued on next page)*

*(continued from previous page)*

| Task | Trial | Base policy | RL only | | | Residual | | |
|---|---|---|---|---|---|---|---|---|
| | | | TD3 | SAC | PPO | TD3 | SAC | PPO |
| | 10 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.5000 |
| OOD | 11 | 0.2500 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.5000 |
| | 12 | 0.7500 | 0.7500 | 0.7500 | 0.7500 | 1.0000 | 0.7500 | 1.0000 |
| | 13 | 0.5000 | 0.5000 | 0.5000 | 0.5000 | 0.7500 | 0.7500 | 0.7500 |
| | 14 | 0.7500 | 0.7500 | 1.0000 | 0.7500 | 1.0000 | 1.0000 | 1.0000 |
| | 15 | 0.5000 | 0.5000 | 0.5000 | 0.5000 | 0.7500 | 0.7500 | 0.7500 |
| | 16 | 0.7500 | 0.7500 | 0.7500 | 0.5000 | 1.0000 | 1.0000 | 1.0000 |
| | 17 | 0.5000 | 0.5000 | 0.5000 | 0.5000 | 0.7500 | 0.7500 | 0.7500 |
| | 18 | 0.7500 | 0.7500 | 0.7500 | 0.7500 | 1.0000 | 0.7500 | 1.0000 |
| | 19 | 0.5000 | 0.5000 | 0.5000 | 0.5000 | 0.7500 | 0.7500 | 0.7500 |
| | 20 | 0.7500 | 0.7500 | 1.0000 | 0.7500 | 1.0000 | 1.0000 | 1.0000 |
| | Mean progress $MP$ | 0.5000 | 0.3625 | 0.4500 | 0.3375 | 0.6750 | 0.6000 | 0.6750 |
| | Generalization $J_{\text{gen}}$ | 0.6250 | 0.6250 | 0.6750 | 0.5750 | 0.8750 | 0.8000 | 0.8750 |
| | Transfer gap $\Delta_{\text{gen}}$ | – | – | – | – | 0.2500 | 0.1750 | 0.2500 |
| Task 3: `bus-table-medium-v1` | | | | | | | | |
| ID | 1 | 0.2500 | 0.0000 | 0.2500 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |
| | 2 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.5000 |
| | 3 | 0.2500 | 0.0000 | 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |
| | 4 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.7500 | 0.5000 |
| | 5 | 0.2500 | 0.0000 | 0.2500 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |
| DP | 6 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.5000 |
| | 7 | 0.2500 | 0.0000 | 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |
| | 8 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.7500 | 0.5000 |
| | 9 | 0.2500 | 0.0000 | 0.2500 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |
| | 10 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.5000 |
| OOD | 11 | 0.2500 | 0.0000 | 0.0000 | 0.0000 | 0.5000 | 0.2500 | 0.5000 |
| | 12 | 0.2500 | 0.2500 | 0.2500 | 0.2500 | 0.7500 | 0.7500 | 0.7500 |
| | 13 | 0.0000 | 0.0000 | 0.2500 | 0.0000 | 0.5000 | 0.5000 | 0.5000 |
| | 14 | 0.2500 | 0.2500 | 0.5000 | 0.2500 | 0.7500 | 0.7500 | 0.7500 |
| | 15 | 0.0000 | 0.0000 | 0.2500 | 0.0000 | 0.5000 | 0.5000 | 0.5000 |
| | 16 | 0.2500 | 0.2500 | 0.0000 | 0.0000 | 0.5000 | 0.5000 | 0.5000 |
| | 17 | 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.5000 |
| | 18 | 0.2500 | 0.2500 | 0.2500 | 0.2500 | 0.7500 | 0.7500 | 0.7500 |
| | 19 | 0.0000 | 0.0000 | 0.2500 | 0.0000 | 0.5000 | 0.5000 | 0.5000 |
| | 20 | 0.2500 | 0.2500 | 0.5000 | 0.2500 | 0.7500 | 0.7500 | 0.7500 |
| | Mean progress $MP$ | 0.3125 | 0.1875 | 0.3125 | 0.1750 | 0.5500 | 0.5375 | 0.5250 |
| | Generalization $J_{\text{gen}}$ | 0.2500 | 0.1500 | 0.2750 | 0.1250 | 0.6000 | 0.6000 | 0.6000 |
| | Transfer gap $\Delta_{\text{gen}}$ | – | – | – | – | 0.3500 | 0.4500 | 0.3500 |
| Task 4: `bus-table-hard-v1` | | | | | | | | |
| ID | 1 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |
| | 2 | 0.2500 | 0.2500 | 0.2500 | 0.0000 | 0.5000 | 0.5000 | 0.5000 |
| | 3 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |
| | 4 | 0.2500 | 0.0000 | 0.2500 | 0.0000 | 0.5000 | 0.5000 | 0.5000 |

*(continued on next page)*

*(continued from previous page)*

| Task | Trial | Base policy | RL only | | | Residual | | |
|---|---|---|---|---|---|---|---|---|
| | | | TD3 | SAC | PPO | TD3 | SAC | PPO |
| | 5 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |
| DP | 6 | 0.2500 | 0.2500 | 0.2500 | 0.0000 | 0.5000 | 0.5000 | 0.5000 |
| | 7 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |
| | 8 | 0.2500 | 0.0000 | 0.2500 | 0.0000 | 0.5000 | 0.5000 | 0.5000 |
| | 9 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.2500 | 0.2500 | 0.2500 |
| | 10 | 0.2500 | 0.2500 | 0.2500 | 0.0000 | 0.5000 | 0.5000 | 0.5000 |
| OOD | 11 | 0.0000 | 0.0000 | 0.2500 | 0.0000 | 0.5000 | 0.5000 | 0.5000 |
| | 12 | 0.0000 | 0.2500 | 0.0000 | 0.2500 | 0.5000 | 0.5000 | 0.5000 |
| | 13 | 0.2500 | 0.0000 | 0.2500 | 0.0000 | 0.7500 | 0.7500 | 0.7500 |
| | 14 | 0.0000 | 0.2500 | 0.0000 | 0.2500 | 0.5000 | 0.7500 | 0.5000 |
| | 15 | 0.2500 | 0.0000 | 0.2500 | 0.0000 | 0.7500 | 0.7500 | 0.7500 |
| | 16 | 0.0000 | 0.2500 | 0.0000 | 0.2500 | 0.5000 | 0.5000 | 0.5000 |
| | 17 | 0.2500 | 0.0000 | 0.2500 | 0.0000 | 0.7500 | 0.7500 | 0.7500 |
| | 18 | 0.0000 | 0.2500 | 0.0000 | 0.2500 | 0.5000 | 0.7500 | 0.5000 |
| | 19 | 0.2500 | 0.0000 | 0.2500 | 0.0000 | 0.7500 | 0.7500 | 0.7500 |
| | 20 | 0.0000 | 0.2500 | 0.0000 | 0.2500 | 0.5000 | 0.5000 | 0.5000 |
| Mean progress $MP$ | | 0.2188 | 0.1562 | 0.2188 | 0.0625 | 0.5625 | 0.5312 | 0.5312 |
| Generalization $J_{\text{gen}}$ | | 0.1250 | 0.1250 | 0.2250 | 0.1250 | 0.6000 | 0.6250 | 0.6000 |
| Transfer gap $\Delta_{\text{gen}}$ | | – | – | – | – | 0.4750 | 0.5000 | 0.4750 |
| Task 5: `close-bottle-lid-v1` | | | | | | | | |
| ID | 1 | 0.7500 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.7500 |
| | 2 | 0.7500 | 0.2500 | 0.5000 | 0.2500 | 0.7500 | 0.7500 | 0.7500 |
| | 3 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.7500 |
| | 4 | 0.7500 | 0.2500 | 0.5000 | 0.2500 | 0.7500 | 0.7500 | 0.7500 |
| | 5 | 0.7500 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.7500 |
| DP | 6 | 0.7500 | 0.2500 | 0.5000 | 0.2500 | 0.7500 | 0.7500 | 0.7500 |
| | 7 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.7500 |
| | 8 | 0.7500 | 0.2500 | 0.5000 | 0.2500 | 0.7500 | 0.7500 | 0.7500 |
| | 9 | 0.7500 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.7500 |
| | 10 | 0.5000 | 0.2500 | 0.5000 | 0.2500 | 0.7500 | 0.7500 | 0.7500 |
| OOD | 11 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.7500 |
| | 12 | 0.5000 | 0.5000 | 0.5000 | 0.5000 | 0.7500 | 0.7500 | 0.7500 |
| | 13 | 0.2500 | 0.2500 | 0.5000 | 0.2500 | 0.5000 | 0.7500 | 0.7500 |
| | 14 | 0.5000 | 0.5000 | 0.5000 | 0.5000 | 0.7500 | 0.7500 | 0.7500 |
| | 15 | 0.2500 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.7500 |
| | 16 | 0.5000 | 0.5000 | 0.5000 | 0.5000 | 0.7500 | 0.7500 | 0.7500 |
| | 17 | 0.2500 | 0.2500 | 0.5000 | 0.2500 | 0.5000 | 0.7500 | 0.7500 |
| | 18 | 0.5000 | 0.5000 | 0.5000 | 0.5000 | 0.7500 | 0.7500 | 0.7500 |
| | 19 | 0.2500 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.7500 |
| | 20 | 0.5000 | 0.5000 | 0.5000 | 0.5000 | 0.7500 | 0.7500 | 0.7500 |
| Mean progress $MP$ | | 0.5750 | 0.3125 | 0.4375 | 0.3125 | 0.6250 | 0.6500 | 0.7500 |
| Generalization $J_{\text{gen}}$ | | 0.4000 | 0.3750 | 0.4250 | 0.3750 | 0.6500 | 0.7000 | 0.7500 |
| Transfer gap $\Delta_{\text{gen}}$ | | – | – | – | – | 0.2500 | 0.3000 | 0.3500 |

*(continued on next page)*

*(continued from previous page)*

| Task | Trial | Base policy | RL only | | | Residual | | |
|---|---|---|---|---|---|---|---|---|
| | | | TD3 | SAC | PPO | TD3 | SAC | PPO |
| Task 6: `erase-whiteboard-v1` | | | | | | | | |
| ID | 1 | 1.0000 | 0.7500 | 0.7500 | 0.5000 | 1.0000 | 1.0000 | 1.0000 |
| | 2 | 0.7500 | 0.7500 | 0.7500 | 0.7500 | 1.0000 | 1.0000 | 0.7500 |
| | 3 | 1.0000 | 0.7500 | 1.0000 | 0.7500 | 1.0000 | 1.0000 | 1.0000 |
| | 4 | 0.7500 | 0.7500 | 0.7500 | 0.7500 | 1.0000 | 1.0000 | 0.7500 |
| | 5 | 1.0000 | 0.7500 | 1.0000 | 0.7500 | 1.0000 | 1.0000 | 1.0000 |
| DP | 6 | 0.7500 | 0.7500 | 0.7500 | 0.7500 | 1.0000 | 1.0000 | 0.7500 |
| | 7 | 1.0000 | 0.7500 | 1.0000 | 0.7500 | 1.0000 | 1.0000 | 1.0000 |
| | 8 | 0.7500 | 0.7500 | 0.7500 | 0.7500 | 1.0000 | 1.0000 | 0.7500 |
| | 9 | 1.0000 | 0.7500 | 1.0000 | 0.7500 | 1.0000 | 1.0000 | 1.0000 |
| | 10 | 0.7500 | 0.7500 | 0.7500 | 0.7500 | 1.0000 | 1.0000 | 0.7500 |
| OOD | 11 | 0.7500 | 0.5000 | 0.7500 | 0.5000 | 1.0000 | 1.0000 | 1.0000 |
| | 12 | 0.7500 | 0.7500 | 0.7500 | 0.7500 | 0.7500 | 1.0000 | 0.7500 |
| | 13 | 0.5000 | 0.5000 | 0.7500 | 0.5000 | 1.0000 | 1.0000 | 1.0000 |
| | 14 | 0.7500 | 0.7500 | 1.0000 | 0.7500 | 1.0000 | 1.0000 | 1.0000 |
| | 15 | 0.7500 | 0.5000 | 0.7500 | 0.5000 | 0.7500 | 1.0000 | 1.0000 |
| | 16 | 0.7500 | 0.7500 | 0.7500 | 0.5000 | 1.0000 | 1.0000 | 1.0000 |
| | 17 | 0.5000 | 0.5000 | 0.5000 | 0.5000 | 1.0000 | 1.0000 | 1.0000 |
| | 18 | 0.7500 | 0.7500 | 0.7500 | 0.7500 | 1.0000 | 1.0000 | 0.7500 |
| | 19 | 0.7500 | 0.5000 | 1.0000 | 0.5000 | 0.7500 | 1.0000 | 1.0000 |
| | 20 | 0.7500 | 0.7500 | 0.7500 | 0.7500 | 1.0000 | 1.0000 | 1.0000 |
| Mean progress $MP$ | | 0.8125 | 0.6250 | 0.7875 | 0.6000 | 0.9625 | 1.0000 | 0.9625 |
| Generalization $J_{\text{gen}}$ | | 0.7000 | 0.6250 | 0.7750 | 0.6000 | 0.9250 | 1.0000 | 0.9500 |
| Transfer gap $\Delta_{\text{gen}}$ | | – | – | – | – | 0.2250 | 0.3000 | 0.2500 |
| Task 7: `close-french-press-v1` | | | | | | | | |
| ID | 1 | 0.7500 | 0.0000 | 0.2500 | 0.0000 | 0.5000 | 0.7500 | 0.5000 |
| | 2 | 0.5000 | 0.2500 | 0.0000 | 0.2500 | 0.7500 | 0.7500 | 0.7500 |
| | 3 | 0.7500 | 0.0000 | 0.2500 | 0.0000 | 0.5000 | 0.7500 | 0.5000 |
| | 4 | 0.5000 | 0.2500 | 0.5000 | 0.2500 | 0.7500 | 1.0000 | 0.7500 |
| | 5 | 0.7500 | 0.0000 | 0.0000 | 0.0000 | 0.7500 | 0.7500 | 0.7500 |
| DP | 6 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.5000 | 0.7500 | 0.5000 |
| | 7 | 0.7500 | 0.0000 | 0.2500 | 0.0000 | 0.7500 | 0.7500 | 0.7500 |
| | 8 | 0.5000 | 0.2500 | 0.0000 | 0.2500 | 0.5000 | 0.7500 | 0.5000 |
| | 9 | 0.7500 | 0.0000 | 0.5000 | 0.0000 | 0.7500 | 1.0000 | 0.7500 |
| | 10 | 0.5000 | 0.2500 | 0.2500 | 0.2500 | 0.7500 | 0.7500 | 0.7500 |
| OOD | 11 | 0.2500 | 0.0000 | 0.0000 | 0.0000 | 0.5000 | 0.7500 | 0.5000 |
| | 12 | 0.0000 | 0.2500 | 0.2500 | 0.2500 | 0.7500 | 0.7500 | 0.7500 |
| | 13 | 0.2500 | 0.0000 | 0.2500 | 0.0000 | 0.5000 | 0.5000 | 0.5000 |
| | 14 | 0.0000 | 0.2500 | 0.0000 | 0.2500 | 0.5000 | 0.7500 | 0.5000 |
| | 15 | 0.2500 | 0.0000 | 0.5000 | 0.2500 | 0.7500 | 0.7500 | 0.7500 |
| | 16 | 0.0000 | 0.2500 | 0.2500 | 0.0000 | 0.5000 | 0.5000 | 0.5000 |
| | 17 | 0.2500 | 0.0000 | 0.0000 | 0.2500 | 0.5000 | 0.7500 | 0.5000 |
| | 18 | 0.0000 | 0.2500 | 0.2500 | 0.0000 | 0.7500 | 0.7500 | 0.7500 |

*(continued on next page)*

*(continued from previous page)*

| Task | Trial | Base policy | RL only | | | Residual | | |
|---|---|---|---|---|---|---|---|---|
| | | | TD3 | SAC | PPO | TD3 | SAC | PPO |
| | 19 | 0.2500 | 0.0000 | 0.2500 | 0.2500 | 0.5000 | 0.5000 | 0.5000 |
| | 20 | 0.0000 | 0.2500 | 0.0000 | 0.0000 | 0.7500 | 0.7500 | 0.7500 |
| Mean progress $MP$ | | 0.3750 | 0.1250 | 0.2000 | 0.1250 | 0.6250 | 0.7375 | 0.6250 |
| Generalization $J_{\text{gen}}$ | | 0.1250 | 0.1250 | 0.1750 | 0.1250 | 0.6000 | 0.6750 | 0.6000 |
| Transfer gap $\Delta_{\text{gen}}$ | | – | – | – | – | 0.4750 | 0.5500 | 0.4750 |

## A.3 SO-101 BILL OF MATERIALS

The bill of materials in **??** provides a consolidated overview of all hardware components required to assemble the two-arm SO-101 setup, including servos, motor control boards, cabling, power supplies, mounting hardware, and tools.

**Table A.2.** Bill of materials for the SO-101 leader and follower robotic arms (two-arm setup).

| Part | Qty |
|---|---|
| STS3215 Servo 7.4V, 1/345 gear (C001) | 7 |
| STS3215 Servo 7.4V, 1/191 gear (C044) | 2 |
| STS3215 Servo 7.4V, 1/147 gear (C046) | 3 |
| Motor control board | 2 |
| USB-C cable (pack of 2) | 1 |
| Power supply | 2 |
| Table clamp (pack of 4) | 1 |
| Screwdriver set | 1 |