Coherent Soft Imitation Learning for Vision-Language-Action models

Coherent Soft Imitation Learning für große Sprach-Vision-Handlungs Modelle
Bachelor thesis by Christian Scherer
Date of submission: November 17, 2025

1. Review: Theo Gruner
2. Review: Daniel Palenicek
3. Review: Jan Peters
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Coherent Soft Imitation Learning for Vision-Language-Action models
Coherent Soft Imitation Learning für große Sprach-Vision-Handlungs Modelle

Bachelor thesis by Christian Scherer

Date of submission: November 17, 2025

Darmstadt

Abstract

This thesis investigates the fine-tuning of Vision-Language-Action Models (VLAs), which require additional adaptation after their pre-training to achieve high success rates for out-of-distribution tasks. Effectively fine-tuning these foundation models is important, as it enables leveraging the vast knowledge acquired during pre-training to solve complex tasks without the often prohibitive cost involved with collecting the required data for training models from scratch. Current literature either relies solely on collecting large amounts of demonstrations for fine-tuning VLAs via Behavioural Cloning (BC) or uses handcrafted rewards or human intervention to fine-tune VLAs via Reinforcement Learning (RL). BC-based methods do not teach the model to recover from suboptimal states, while handcrafted rewards and human-intervention-based methods do not scale well to complex tasks. This thesis proposes an approach for fine-tuning VLAs by training a residual using Coherent Soft Imitation Learning (CSIL). CSIL leverages a policy trained via BC for acting in the environment and producing reward signals, which enables us to train a residual that matches or improves the base VLA performance and then increase the robustness of the residual using additional online interactions without needing to specify a task-specific reward function. Crucially, this research also uncovers a fundamental challenge in applying CSIL to VLAs embeddings: the high-dimensional embeddings from the VLA lead to a collapse of the model's uncertainty estimates, undermining CSIL's reward mechanism.

Contents

Abbreviations, Symbols and Operators

List of Abbreviations

VLA	Vision-Language-Action Model
VQA	Visual Question-Answering
VLM	Vision-Language Model
CSIL	Coherent Soft Imitation Learning
RL	Reinforcement Learning
IRL	Inverse Reinforcement Learning
BC	Behavioural Cloning
LLM	Large Language Model
GRP	Generalist Robot Policy
SAC	Soft Actor-Critic
SACFD	Soft Actor-Critic from Demonstrations
LORA	Low-Rank Adaptation
MLP	Multi-layer Perceptron
MDP	Markov Decision Process

1. Introduction

The emergence of large-scale, pre-trained foundation models is a promising step towards Generalist Robot Policies (GRPS), yet sample-efficiently adapting these policies to solve complex tasks remains a significant challenge. Standard fine-tuning methods like Behavioural Cloning (BC) are brittle, while Reinforcement Learning (RL) is more robust but often requires prohibitive sample sizes. This thesis addresses this critical gap by investigating a hybrid fine-tuning approach in the context of Vision-Language-Action Models (VLAS) that combines the strengths of both paradigms to achieve robust and sample-efficient policy refinement.

1.1. Motivation

Automating tasks lies at the core of robotics. While industrial automation has mastered repetitive, large-scale manufacturing, a vast range of tasks that may seem repetitive to the human eye remain beyond the reach of traditional programming. Training a policy from scratch to solve a single, complex task that may involve many different objects or environments requires large amounts of demonstrations for the needed generalisation capabilities to emerge. It is therefore desirable to have foundation models that reduce the required number of samples to solve such a task by offering a strong basis for the policy to start from.

In recent years, Natural Language Processing and Computer Vision have seen large breakthroughs due to foundation models [18, 9, 33]. In both cases, the recipe for success included large transformer networks trained on diverse, large-scale datasets [32, 10]. Transferring this recipe to robotics has already shown impressive results in terms of generalisation capabilities as well as few-shot performance, though often fine-tuning these foundation models for complex tasks still requires a significant amount of manually collected data [4].

In part, this could be attributed to the method used for fine-tuning such models, as the canonical way for fine-tuning is BC, which assumes the demonstrations were created by an optimal policy and tries to mimic the optimal policy's behaviour

represented in the dataset. During rollouts, matching the dataset's actions works well for in-distribution states, but often becomes brittle for out-of-distribution states, as the model has to infer the action, yet has never seen suboptimal states or how to recover from them. To counteract BC's inability to teach the model to recover from suboptimal states, one has to present a large part of the relevant state space in the dataset to enable the model to infer the correct actions [34].

RL offers a compelling alternative, as it enables a policy to learn from its own experience through trial and error, thereby discovering recovery behaviours by exploring the state-space [29]. The process of designing a reward function that produces the desired behaviour is complex and often labour-intensive, as the implications of a specific reward function design usually only become apparent after applying it in practice. As robotic manipulation can be split into multiple phases, a reward function that defines separate reward functions for each phase may simplify the task by enabling the reward engineer to focus on only specific parts of the desired behaviour. Yet, the transition between different stages often leads to sharp edges in the value function, which is detrimental to the downstream performance. Relying on sparse rewards, such as a simple signal upon task completion, suffers from immense sample inefficiency, especially for increasingly complex tasks, as a vast majority of states do not offer any reward signal to learn from, significantly restricting its real-world applicability [15, 17]. On the other hand, learning reward functions often involves a careful manual extraction of features, such as the end-effector velocity approaching zero or changing its gripper openness. Such features often do not scale well to complex tasks, due to a lack of richness in their encoding of the task.

Thus, there is a need for a fine-tuning approach that is robust to suboptimal states, sample-efficient, but not dependent on handcrafted reward functions or reward functions trained on handcrafted features. Coherent Soft Imitation Learning (CSIL) [38] enables us to use the good initialisation of BC while also training the policy to recover from suboptimal skills using a dense reward function that can be extracted directly from the pre-trained policy, combining the benefits of BC and RL.

1.2. Proposed Approach

This thesis introduces an approach for fine-tuning VLAs, designed to combine the sample efficiency of BC with the robustness of RL. Our approach is built upon two core technical components:

1. CSIL provides a method for leveraging a pre-trained policy as an actor in the environment as the reward function for teaching the policy robustness using RL.

2. Policy Residuals enable model-agnostic, parameter-efficient fine-tuning, by adding a learnable delta on top of the action predicted by the base policy.

The primary innovation of this work lies in applying and scaling CSIL, previously only validated on low-dimensional state information as well as on an image-based policy trained from scratch, to the domain of large, pre-trained VLAs. A part of current literature transfers the paradigm of fine-tuning Large Language Models (LLMs) to VLAs often by simply applying BC [22]. Such approaches overlook a fundamental advantage that the robotic manipulation domain provides: the availability of automatically verifiable task completion. The language domain usually requires human feedback for receiving sparse rewards, i.e. a metric to signal to the model which output was good or bad. In comparison, the robotic manipulation domain usually has a clear goal state, e.g. the cube is lifted off the ground or the apple is placed on top of the plate. Therefore, VLAs are even better suited for RL-based fine-tuning than LLMs, where retrieving a reward is non-trivial, yet has already shown impressive results [14].

The main contributions of this thesis are:

1. We introduce a novel approach for integrating CSIL and residual learning into the domain of VLAs. This contribution is significant, as it provides a pathway to adapt VLAs without the need for extensive data collection or manual reward design.

2. We present a comprehensive experimental validation of our approach, tackling additional, more complex tasks in the image-based setting than evaluated in Watson, Huang, and Heess (2023). This validation demonstrates the promising capabilities of VLAs and our approach, while highlighting current limitations in more complex environments.

3. We provide a detailed analysis of a challenge that arises when applying CSIL to high-dimensional VLA embeddings: the degradation of the uncertainty quantification, used in CSIL's reward function. This analysis identifies a fundamental bottleneck for the applicability of our approach and informs future research aimed at finding a more robust solution in the high-dimensional input setting.

The remainder of this thesis is organised as follows: We first review relevant literature on VLAs and their fine-tuning. Then, we establish the theoretical foundations underpinning our approach, providing an introduction to Entropy-Regularised Reinforcement Learning and Coherent Soft Imitation Learning. Following the foundational chapter, the architecture and algorithm of our proposed fine-tuning approach are detailed. We then describe the experimental setup used for our evaluations and subsequently present and discuss the experimental results, as well as investigate a current key issue as described in the main contributions. The thesis is concluded by summarising the key findings, addressing limitations, and proposing future work.

2. Related Work

Generalist Robot Policies have long been a dream in robotics, concerned with finding a policy that solves a broad range of tasks, even those not seen before. VLAs are a promising step towards GRPs, having already shown initial signs of being able to transfer task knowledge between embodiments and environments [4, 7, 8]. This chapter introduces relevant literature surrounding VLAs and their fine-tuning process.

2.1. Vision-Language-Action Models

The canonical approach to solving a task using Robot Learning is to collect demonstration data, pick an architecture suitable for the task and then train a model of the specified architecture using BC. This approach works well for simple tasks and those where a large amount of expert demonstration data is available. However, BC often reaches its limits for long-horizon or otherwise complex tasks. A GRP may be a solution to this problem, as it would be able to solve a broad range of tasks and even generalise from seen tasks, environments and embodiments to unseen ones. Yet training a GRP requires robustness towards covariate shift, enabling solving known and unknown tasks in unknown environments and the ability to combine previously learnt information to apply it to new problems.

VLAs are an attempt at closing the gap between the current state-of-the-art and a true GRP. VLAs generally receive the current state of the environment in the form of images and a specification of the task in natural language. Optionally, end-effector pose information and other input modalities may also be provided.

Early influential work, such as RT-1 [7], demonstrated the power of the transformer architecture for this purpose, training a model on a large dataset of kitchen-based tasks. The subsequent model, RT-2 [8], advanced this concept by leveraging a pre-trained Vision-Language Model (VLM) as its foundation, reframing action generation as a Visual Question-Answering (VQA) problem. This allowed the policy to inherit the rich semantic understanding of the world embedded in

the VLM's web-scale pre-training data, leading to improved generalisation and the ability to solve a variety of seen and unseen tasks.

While both RT-1 and RT-2 showed promising results in terms of overall performance and generalisation, their capabilities were often constrained by the diversity of their training data. Open X-Embodiment [12] addressed this issue by enlargening the training mixture significantly, combining datasets from 21 institutions. By simply training the same architectures used for RT-1 and RT-2 [7, 8] on a much larger and more diverse dataset, the authors demonstrated improvements on seen tasks as well as in generalisation capabilities. Further, open-sourcing the dataset and expanding the collection since its initial release has accelerated research into VLAs.

Though Open X-Embodiment [12] made their dataset and the weights for their smaller RT-1-X model publicly available, crucial parts, such as a data loader as well as the training pipeline, were not released. Their more advanced model, RT-2-X, was also not released. Octo [31] a VLA similar in size to RT-1-X used a transformer backbone and a diffusion head and addressed this gap by releasing its data loader, training pipeline and weights. OpenVLA [23] addressed the gap of a missing open-source VLA that uses a pre-trained VLM by fine-tuning PrismaticVLM [21] using data from Open X-Embodiment and subsequently releasing the weights and code.

A key architectural limitation of VQA-based models like RT-2 and OpenVLA is their reliance on autoregressively generating action tokens, a process that can be computationally slow and limit real-time performance. π_0 [4] addresses this issue by training a model that produces multiple actions per inference call. π_0 is a state-of-the-art model that relies on PaliGemma [3] as its backbone and uses Flow Matching [24] for generating action chunks. It decouples the high-level semantic understanding of vision and language information from the low-level action generation by separating the stages into 2 models built on the PaliGemma architecture. The first model is a pre-trained VLM, responsible for encoding the vision and language information, while the second set of weights is trained from scratch and is responsible for encoding proprioceptive information to iteratively denoise an initially randomly sampled action chunk using flow matching. The output tokens from the first set of weights are used for attending to the tokens from the second set of weights using cross-attention. As flow matching requires multiple forward passes, π_0's architecture choice increases inference speed significantly by enabling the output of the first set of weights to be cached between flow matching passes.

In summary, VLAs have transferred the paradigm of training large transformer-based models with large, diverse datasets from LLMs and have gone from reframing end-effector control as a VQA problem [8, 23] to introducing a more hierarchical approach, enabling the introduction of parallel action encoding and even the prediction of entire action chunks [4].

2.2. Fine-tuning Vision-Language-Action Models

LLMs and vision foundation models have demonstrated not only the ability to generalise broadly after large-scale pre-training, but also their potential for achieving impressive performance with RL-based fine-tuning [14]. VLAs, similarly, have shown impressive zero-shot generalisation across diverse robotic tasks after pre-training [4, 8, 31]. Though they solve difficult and even unseen tasks in some cases, achieving high performance on long-horizon or otherwise complex tasks necessitates task-specific fine-tuning. Current literature on fine-tuning VLAs is primarily divided between two paradigms: imitation via Behavioural Cloning and adaptation via Reinforcement Learning.

BC-based fine-tuning is conceptually simple and widely adopted. It intends to teach a policy to solve a task by mimicking behaviour represented in expert demonstrations. Despite its simplicity, it already enables VLAs to solve complex tasks [4, 22]. However, it requires substantial amounts of high-quality expert demonstrations, which are costly to obtain, and lacks mechanisms for teaching corrective or recovery behaviours [34].

RL-based fine-tuning, on the other hand, enables the policies to become much more robust to out-of-distribution states. VLA-RL [26] showed RL's potential for fine-tuning VLAs by applying PPO [37] to OpenVLA [23] utilising handcrafted features for learning a reward function. A solution that does not scale easily to more complex problems.

An alternative strategy is to use human-in-the-loop methods, where a human provides real-time interventions or preferences to guide the policy, as seen in approaches like HAPO [39] and ConRFT [11]. While effective, these methods are labour-intensive and difficult to deploy at scale.

A parallel challenge in fine-tuning is computational efficiency. Updating the weights of a multi-billion-parameter VLA is resource-intensive. Consequently, most approaches employ parameter-efficient fine-tuning techniques. The most common of these is Low-Rank Adaptation (LoRA) [20], which freezes the base model's weights and trains a much smaller set of new weights. Though it reduces the

number of parameters significantly, it can still require substantial computational resources.

Residual learning offers a compelling alternative for parameter-efficient fine-tuning. This approach also freezes the large base model but trains a separate, much smaller model that learns to predict a corrective residual to the base policy's action. Base models offer a good foundation for solving the task, but often fail in long-horizon or complex tasks due to compounding errors. Residuals can minimise these compounding errors by being trained specifically on the task. Recent work [1, 40] has demonstrated the efficacy of applying residual learning to large policies, using RL with sparse rewards to train the residual component.

The approaches discussed demonstrate a variety of methods for fine-tuning VLAs from BC-based methods to RL-based methods, using handcrafted reward signals, human intervention for relabeling episodes or residual components. The proposed method combines a residual component initialised with BC with a dense reward function that can be extracted from the policy without requiring further human intervention or reward engineering.

3. Foundations

3.1. Entropy-Regularised Reinforcement Learning

The conventional framework for Reinforcement Learning is the Markov Decision Process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}$ is set of states, $\mathcal{A}$ a set of actions, $p(\mathbf{s}_{t+1}|\ \mathbf{s}_t, \mathbf{a}_t)$ a function specifying the transition probabilities and a reward function $r(\mathbf{s}, \mathbf{a})$. In a MDP, the agent receives the current state $\mathbf{s}_t$ and chooses an action $\mathbf{a}_t$. The environment then steps into a new state $\mathbf{s}_{t+1}$ depending on the transition probabilities $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. Depending on the formalisation of the MDP, the agent either receives a reward for being in a state or receives it for taking a specific action in a state. In classical, finite-horizon RL, the objective is to learn a policy that maximises the cumulative reward, i.e.

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}\left[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

Greedily maximising the reward, however, is prone to premature convergence of the policy to local minima. In such scenarios, the resulting policy may prematurely collapse into a limited mode of behaviour, thereby restricting exploration and impeding the subsequent discovery of higher-value state–action pairs and more optimal policies. Entropy-Regularised RL addresses this challenge by expanding the classic RL objective of maximising the reward by maximising the sum of the reward and the entropy, where the entropy H is weighted by a temperature α, leading to the following optimal policy definition:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \underset{\tau\sim\pi}{\mathbb{E}}\left[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t) + \alpha H(\pi(\cdot|\mathbf{s}_t))\right].$$

The entropy is defined as $H(\pi(\cdot|s)) = \mathbb{E}_{a\sim\pi(\cdot|s)}[-\log \pi(a|s)]$.

This entropy bonus incentivises the policy to maintain a higher entropy in its action selection. Maintaining higher policy entropy encourages broader explo-

ration of the state space, thereby increasing the likelihood that the agent will discover a more globally optimal policy.

3.2. Coherent Soft Imitation Learning

CSIL [38] is a method for training a policy with expert demonstrations and additional data from interactions via BC and successive Inverse Reinforcement Learning (IRL). CSIL's primary goal is to increase the robustness of a policy by teaching it to transition from out-of-distribution states to in-distribution states. CSIL achieves this by formulating the following reward function

$$r(\mathbf{s}, \mathbf{a}) = \alpha \log \frac{\pi_\alpha(\mathbf{a}|\mathbf{s})}{p(\mathbf{a}|\mathbf{s})}. \tag{3.1}$$

The reward function is based on the difference in the log-likelihoods of the policy, pre-trained using BC, and a uniform prior. The ratio is further scaled by a tunable parameter α. The authors call this connection between the policy and the reward function *coherence*.

The reward function specified in (3.1) should reward state-action pairs for which the state and action are in-distribution, should penalise state-action pairs where the state is in-distribution and the action is not and should return a reward of 0 if the state is out-of-distribution, independent of the action. To achieve the desired reward function behaviour, it is required that π_α models the data $\mathcal{D}$ when in-distribution and models the uniform prior otherwise.

In practice, π_α is usually a function approximator. As function approximators represent correlations between states, updating their approximation for a subset of states usually changes the approximation for a large set of other states as well. As the reward function's desired behaviour requires π_α to match the prior for out-of-distribution states, the authors introduce an architecture that transforms learning into a stationary process.

Definition 1 *(Stationary process, [13, 38]). A process $f : \mathcal{X} \to \mathcal{Y}$ is stationary if its joint distribution in $\mathbf{y} \in \mathcal{Y}, p(\mathbf{y}_1, ..., \mathbf{y}_n)$, is constant w.r.t. $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathcal{X}$ and all $n \in \mathbb{N}_{>0}$*

The authors use results that show a wide final layer with a periodic activation function induces stationarity [30, 38].

They define the policy as $a = \tanh(z(\mathbf{s}))$, where $z(\mathbf{s}) = \mathbf{W}\phi(\mathbf{s})$, for which the weights are factorized row-wise $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_{d_\alpha}]^\mathsf{T}, \mathbf{w}_i = \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ to define a Gaussian Process with independent actions.

By using change-of-variables as in SAC [19], the authors express the policy per-action as

$$q(\mathbf{a}_i|\mathbf{s}) = \mathcal{N}\left(z_i; \boldsymbol{\mu}_i^\mathsf{T}\phi(\mathbf{s}), \phi(\mathbf{s})^\mathsf{T}\boldsymbol{\Sigma}_i\phi(\mathbf{s})\right) \cdot \left|\det\left(\frac{da_i}{dz_i}\right)\right|^{-1}, \phi(\mathbf{s}) = f_{per}(\tilde{\mathbf{W}}\phi_{mlp}(\mathbf{s})),$$

where f_{per} is a periodic activation function [30], the weights $\tilde{\mathbf{W}}$ are drawn from a distribution, e.g. a Gaussian that characterises the stochastic process, and ϕ_{mlp} is an arbitrary MLP [38].
They refer to this heteroscedastic stationary model as HETSTAT. The critic used for SAC is then initialised via

$$\tilde{Q}_1(\mathbf{s}, \mathbf{a}) = \tilde{r}_{\boldsymbol{\theta}_1}(\mathbf{s}, \mathbf{a}) + \gamma\big(\tilde{Q}_1(\mathbf{s}', \mathbf{a}') - \alpha(\log q_{\boldsymbol{\theta}_1}(\mathbf{a}' \mid \mathbf{s}') - \log \pi(\mathbf{a}' \mid \mathbf{s}'))\big),$$

where $\mathbf{s}, \mathbf{a}, \mathbf{s}', \mathbf{a}' \sim \mathcal{D}$ and $(\mathbf{s}', \mathbf{a}')$ is the successor state-action pair to $(\mathbf{s}, \mathbf{a})$. The reward's weights are then separated from the policy's weights such that both can be updated independently, and the policy and critic are updated using a Soft Policy Iteration method; in practice, Soft Actor-Critic (SAC) is used.

To counteract the distribution shift in the policy that occurs during training, CSIL refines the coherent reward during RL. This is achieved using the following objective

$$\mathcal{J}_r(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[\alpha \log \frac{q_{\boldsymbol{\theta}(\mathbf{a}|\mathbf{s})}}{p(\mathbf{a}|\mathbf{s})}\right] - \mathbb{E}_{\mathbf{s},\mathbf{a}\sim p_\pi}\left[\alpha \log \frac{q_{\boldsymbol{\theta}(\mathbf{a}|\mathbf{s})}}{p(\mathbf{a}|\mathbf{s})}\right].$$

Following Schulman (2020), the authors construct an unbiased, positively-constrained KL estimator, as the second term of the objective can be seen as a Monte Carlo estimation of the relative KL divergence between q and p if π is $q_{\boldsymbol{\theta}(\mathbf{a}|\mathbf{s})}$ [38]. The estimator results in replacing the second term from the reward refinement objective with

$$\mathbb{E}_{\mathbf{s},\mathbf{a}\sim p_\pi}\left[r_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) - 1 + \exp(-r_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}))\right].$$

In practice, the samples are taken from a replay buffer of online and offline data, so the action samples are not taken from the same distribution as $q_{\boldsymbol{\theta}}$. The objective is still, however, effective for reward regularisation. [38]

Positive rewards lead to behaviour that tries to stay in high-reward states for as long as possible. For robotic manipulation tasks, this leads to behaviour, where the policy, e.g. closely hovers around the object it should grasp or move to maximise its reward while not actually completing the task, as that would end the episode and therefore diminish the cumulative episode return. To prevent this behaviour, the coherent reward can be bounded and its value subsequently shifted by its maximum reward, to ensure the coherent reward is always negative. To achieve this bounding of the maximum reward, the authors add a small bias term σ^2_{min} to the predictive variance that enables them to define an upper bound for the likelihood. As the $\tanh$ transformation is also bounded in practice to ensure numerical stability, the authors are then able to derive the following upper bound for the reward $r(\mathbf{s}, \mathbf{a})$, given action dimension d_a, a uniform prior across the action range $[-1, 1]^{d_a}$ and $\alpha = d_a^{-1}$

$$r(\mathbf{s}, \mathbf{a}) \leq -0.5 \cdot \log \pi \sigma^2_{min}/2 + \tilde{c},$$

where $\tilde{c}$ depends on the $\tanh$ clipping term [38].

Algorithm 1 Coherent Soft Imitation Learning

Data: Expert demonstrations $\mathcal{D}$, initial temperature α, refinement temperature β, parametric policy $q_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s})$, prior policy $p(\mathbf{a} \mid \mathbf{s})$, regression regularizer Ψ, iterations N

Result: $q_{\boldsymbol{\theta}_N}(\mathbf{a} \mid \mathbf{s})$, matching or improving the initial policy $q_{\boldsymbol{\theta}_1}(\mathbf{a} \mid \mathbf{s})$

Train initial policy from demonstrations, $\boldsymbol{\theta}_1 = \text{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}[\log q_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s}) - \Psi(\boldsymbol{\theta})]$

Define fixed shaped coherent reward, $\tilde{r}_{\boldsymbol{\theta}_1}(\mathbf{s}, \mathbf{a}) = \alpha(\log q_{\boldsymbol{\theta}_1}(\mathbf{a} \mid \mathbf{s}) - \log p(\mathbf{a} \mid \mathbf{s}))$

Choose reference policy $\pi(\mathbf{a} \mid \mathbf{s})$ to be $p(\mathbf{a} \mid \mathbf{s})$ or $q_{\boldsymbol{\theta}_1}(\mathbf{a} \mid \mathbf{s})$ and initialize the shaped critic, $\tilde{Q}_1(\mathbf{s}, \mathbf{a}) = \tilde{r}_{\boldsymbol{\theta}_1}(\mathbf{s}, \mathbf{a}) + \gamma\big(\tilde{Q}_1(\mathbf{s}', \mathbf{a}') - \alpha(\log q_{\boldsymbol{\theta}_1}(\mathbf{a}' \mid \mathbf{s}') - \log \pi(\mathbf{a}' \mid \mathbf{s}'))\big), \quad \mathbf{s}, \mathbf{a}, \mathbf{s}', \mathbf{a}' \sim \mathcal{D}$.

for $n = 2 \rightarrow N$ **do** // fine-tune policy with Reinforcement Learning

Compute $\tilde{Q}_n$ and $q_{\boldsymbol{\theta}_n}$ using soft policy iteration, e.g. SAC, with temperature β.

end for

Taken from "Coherent Soft Imitation Learning" [38]

4. Coherent Soft Imitation Learning for Vision-Language-Action Models

This chapter describes our approach for improving VLA performance by integrating it into CSIL using a residual component. As our approach is model-agnostic, it is well-suited for the fast-changing research landscape of VLAs.

The major architectural change of our approach is the addition of a HETSTAT residual component. This component acts as a second action head, using the embeddings from π_0 in addition to the end-effector pose. Figure 4.1 shows how we retrieve the embeddings from π_0. To combine the action from π_0's action head with the action distribution from the residual component, we shift the mean of the action distribution by the VLA action. To account for the $\tanh$ transformation that is applied to our action distribution, we apply $\operatorname{arctanh}$ to the VLA's action. This results in our final action distribution being defined as

$$q_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s}, \mathbf{a}_{\text{vla}}) = \tanh(\mathbf{z}), \mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{HETSTAT}} + \operatorname{arctanh}(\mathbf{a}_{\text{vla}}), \boldsymbol{\Sigma}_{\text{HETSTAT}})$$

where $\boldsymbol{\mu}_{\text{HETSTAT}}$ and $\boldsymbol{\Sigma}_{\text{HETSTAT}}$ are predicted by the HETSTAT. This formulation allows us to seamlessly integrate the VLA into CSIL's pre-training and RL training by reformulating the reward and critic as

$$\begin{aligned} r(\mathbf{s}, \mathbf{a}_{\text{vla}}, \mathbf{a}) &= \alpha \log \frac{\pi_\alpha(\mathbf{a}|\mathbf{s}, \mathbf{a}_{\text{vla}})}{p(\mathbf{a}|\mathbf{s}, \mathbf{a}_{\text{vla}})} \\ \tilde{Q}_1(\mathbf{s}, \mathbf{a}_{\text{vla}}, \mathbf{a}) &= \tilde{r}_{\boldsymbol{\theta}_1}(\mathbf{s}, \mathbf{a}_{\text{vla}}, \mathbf{a}) + \gamma \tilde{Q}_1(\mathbf{s}', \mathbf{a}'_{\text{vla}}, \mathbf{a}'), \end{aligned}$$

where $\mathbf{s}, \mathbf{a}_{\text{vla}}, \mathbf{a}, \mathbf{s}', \mathbf{a}'_{\text{vla}}, \mathbf{a}' \sim \mathcal{D}$. This formulation keeps the additional implementation complexity low, as only the VLA action additionally needs to be stored in comparison to CSIL.

Additionally, as a second option for combining the VLA action with the HETSTAT action, we propose using the HETSTAT as part of an ensemble,
where $q_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s}, \mathbf{a}_{\text{vla}}) = (\tanh(\mathbf{z}) + \mathbf{a}_{\text{vla}})/2, \mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{HETSTAT}}, \boldsymbol{\Sigma}_{\text{HETSTAT}})$. An ensemble approach allows us to increase the robustness of the model by averaging

over the VLA action and the HETSTAT action without influencing the loss landscape for the HETSTAT.

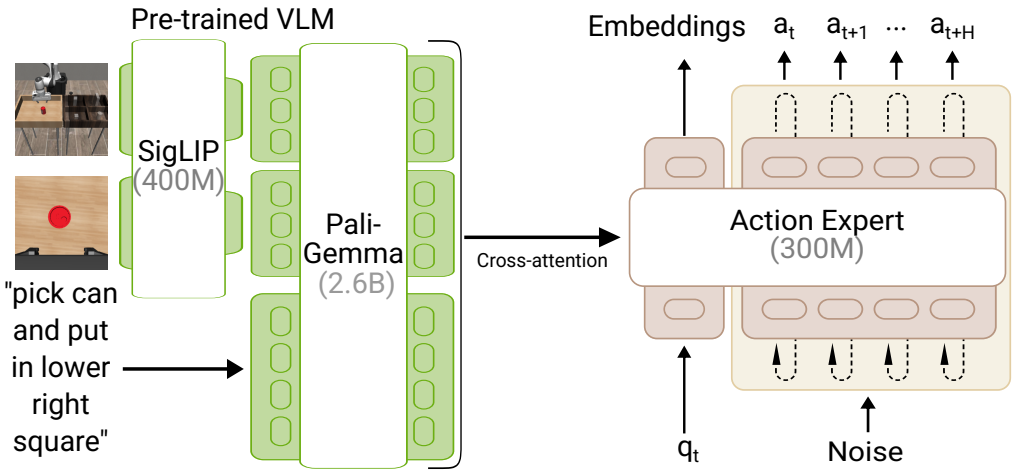


Figure 4.1.: Visualisation of π_0's architecture, adapted from [4]. First Vision and Text data are processed through the pre-trained 3B parameter PaliGemma VLM [3]. Then the current state information and random noise are fed to a second, smaller PaliGemma network. The tokens from the first network are attended to in the second network via cross-attention. π_0 uses Flow Matching [24] to denoise the initial noise to an action chunk. Therefore, the second network is called multiple times to predict a single action chunk. CSIL's policy, reward and critic are trained on the embeddings returned by π_0.

5. Experiments

This chapter introduces the experimental evaluation of applying CSIL on VLAS using a residual component. The experiments are designed to assess the effect of our approach on downstream performance. We begin by explaining implementation specifics, followed by detailing our general experimental setup and the configuration of our experiments.

5.1. Implementation Details

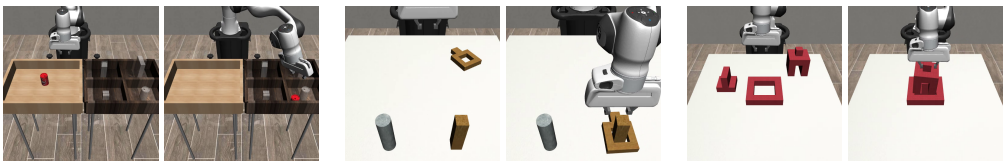
We implemented our system using JAX [6], a high-performance, numerical computation library. Our codebase builds on CSIL's original codebase, though large parts of it have been reimplemented by us, as CSIL's original codebase uses haiku and acme, which both use an outdated JAX version. Our implementation of SAC [19] is built on top of RL-X [5].

5.2. Experimental Setup

To evaluate our method, we conducted experiments on 3 simulated 7-DoF Franka robot environments, 2 from Robosuite [41] and 1 from Mimicgen [27]. We use π_0 [4] as the base VLA for our residual component. We fine-tune π_0 using human-generated expert demonstration data from our environments, as π_0 action normalisation needs to be adjusted to our setup as well as to reduce the real-to-sim gap. In this section, we will introduce the task specifications of our environments, detail the process of fine-tuning π_0 and outline the experiments we conducted.

5.2.1. Environments

In general, for all tasks used in our experiments, the observations provided are an image of the environment, a wrist image and the current end-effector state given



Can Square Assembly

Figure 5.1.: **Tasks.** We evaluate our approach on 3 tasks implemented using Robosuite [41]. The left image for each task depicts the initial state of the environment, while the right image depicts the last frame of a successful episode.

in an eight-dimensional vector. All environments expect a seven-dimensional vector that specifies the change in position, rotation and gripper openness of the end-effector. All environments provide a sparse reward of 1 at successful completion of the episode and 0 otherwise. The episode terminates at successful completion. The tasks are shown in Fig. 5.1. The robots are controlled at a frequency of 20Hz.

Can: The goal of Can is to grab a can placed on the left side of the table and put it in the quadrant that has a grey can inside situated on the right side of the table. The maximum episode length of Can is 400 steps. The can position in the wooden square is randomised between episodes. The target quadrant is always the bottom right.

Square: The goal of Square is to grab a wooden square ring and put it on a stick. The maximum episode length of Square is 400 steps. The ring position is randomised between episodes and the target position is fixed.

Assembly: The goal of Assembly is to assemble a three-piece structure by sequentially picking and placing objects. The maximum episode length of Assembly in our experiments is 800 steps. The initial position of the square in the centre is fixed between episodes, while the positions of the other 2 objects are randomised.

5.2.2. Fine-tuning π_0

π_0 is a state-of-the-art VLA trained on a mix of real-world datasets ranging across multiple embodiments. As π_0 has not been pre-trained on the environments used for evaluation, its normalisation metrics initially do not fit the environments. Additionally, as it is trained on real-world data, it does not perform as well in simulation as it does in the real world due to the real-to-sim gap. To still be able

to evaluate our approach in simulation, we opt for fine-tuning π_0 on the demonstration datasets for the environments used, such that both the normalisation metrics and the real-to-sim gap are addressed. For the Robosuite environments, we made use of Robomimic's [28] proficient-human datasets, and for the Assembly environment, we used a dataset generated using the mimicgen [27] pipeline.

We fine-tune π_0's Vision-Language backbone as well as its action expert using LORA [20]. π_0's training pipeline uses data augmentation to reduce overfitting. It applies colour jitter to all images and additionally randomly crops and rotates images that are not wrist images. To investigate our approach across different ranges of prior task proficiency, we separately fine-tune three π_0 models. To simulate a model that is able to solve the specified tasks with a low proficiency, we fine-tune one model for 10k steps on an unbalanced dataset mix consisting of 200 demonstrations each for the robomimic tasks and 1000 episodes for Assembly. Due to a misconfiguration, the episodes for the Assembly task had a resolution of 84×84 instead of the desired 224×224. In the experiment results, this model is called the suboptimally trained model. To investigate our approach's effectiveness for better adapted models, we fine-tune two π_0 models for 30k steps on a balanced dataset consisting of 200 demonstrations for each environment, where one is fine-tuned using LORA and one is fully fine-tuned.

5.2.3. Evaluating Performance

To evaluate our approach, we run experiments on all 3 environments, each for all 3 of our fine-tuned π_0 versions for 4 seeds each. Due to the misconfiguration of the resolution for the suboptimal model, we do not report its results for Assembly, as they are not representative of the approach. The policy and critic for our approach are trained for 250k RL steps, evaluating the model every 25k steps for 25 episodes. To baseline our performance, we evaluate the performance of both the π_0 version as well as from base π_0 on all 3 environments for 25 episodes over 4 seeds. We further train a residual on the same environments and using the same π_0 versions using Soft Actor-Critic from Demonstrations (SACFD) [35] with a demonstration-to-online-data ratio of 0.2. Due to the poor performance of SACFD, we increased its number of training steps to 500k. We increase the evaluation frequency from every 25k steps to every 50k steps for SACFD due to time constraints. A more comprehensive overview of our hyperparameters is presented in Appendix B.

To ablate the choice of using a residual component, we evaluate the proposed approach's performance when using residual component, when using the HETSTAT

as part of an ensemble and when using the HETSTAT as an action head, where $q_{\boldsymbol{\theta}}(\mathbf{a} \mid \mathbf{s}, \mathbf{a}_{\text{vla}}) = \tanh(\mathbf{z}), \mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{HETSTAT}}, \boldsymbol{\Sigma}_{\text{HETSTAT}})$.

Furthermore, we investigate a current pressing issue regarding a discrepancy between the desired entropy of the policy for online states and the actual entropy of our policy for online states. Initially, we look at the pre-training dynamics of this issue regarding entropy by evaluating our policy during pre-training using a stochastic actor and reporting the mean online entropy. Later on, we repeat our main experiments for the fully fine-tuned π_0 model to investigate whether updating the vision encoder during π_0 pre-training alleviates the issue.

6. Experimental Results

This chapter presents a comprehensive analysis of the experimental results obtained from evaluating our residual IRL approach. We assess the performance of our approach for 3 different environments.

We examine success rates and ablate the architectural choice of training a residual compared to training a simple action head or using the HETSTAT as part of an ensemble. Additionally, we investigate an issue concerning the stationarity of our policies when using VLA embeddings as inputs.

6.1. Performance Comparison

To assess the effectiveness of our approach, we compare it against SACFD [35] and the VLA's base performance for both the suboptimally-trained π_0 model as well as the π_0 model, trained on a balanced dataset and following π_0's default fine-tuning recipe [4].

6.1.1. Suboptimally-trained Model

The experimental data, presented in Figure 6.1, reveals a sharp drop in performance for the SACFD baseline between the first two evaluations. The residual policy is initialised to have a mean of zero in its predicted action distributions and therefore likely does not negatively impact the performance of the VLA too much before starting RL. The initial gradient steps of RL then lead to the HETSTAT's mean deviating from zero. As the residual is not proficient in predicting residual actions, these larger, poorly-formed predictions degrade the overall performance. For Can, SACFD is able to recover its initial performance within around 200k steps and then proceeds to converge towards a success rate of 1.0 slowly. For the more complex Square task, it is unable to recover its initial performance, and only at the very last evaluation does it solve the task in evaluation after starting RL.

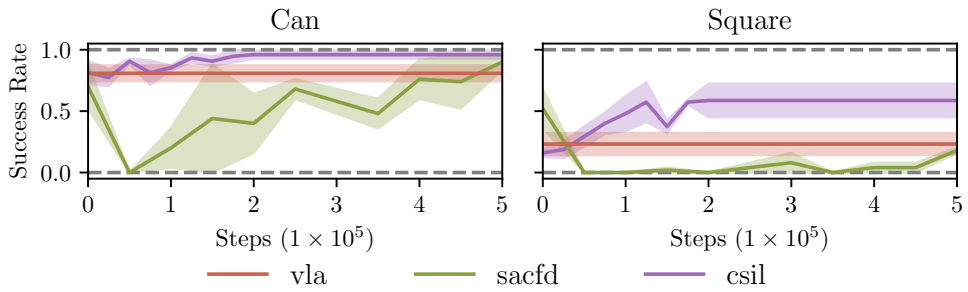


Figure 6.1.: Performance comparison in 2 robotic manipulation environments using the suboptimally-trained model. Success rates for our approach (green), SACFD (purple) and the VLA's base performance (red). Our approach consistently outperforms the baseline, even though SACFD is trained for 2.5 times the steps. Uncertainty intervals depict one standard deviation.

The proposed approach consistently outperforms both SACFD as well as the base performance of the VLA for Can and Square. For Can, our approach consistently surpasses the VLA base performance after 100k steps. For Square, it reaches superior performance within 50k steps and achieves up to 60% success rate after 200k steps compared to the VLA's 25% average success rate. A key advantage is the absence of the initial performance drop observed in the SACFD baseline, which we attribute to the coherence between reward function and policy. In contrast to SACFD, our approach provides a dense reward function, which is simpler to learn from than a sparse reward.

6.1.2. Model Trained on Balanced Dataset

The results in Figure 6.2 show the same sharp drop in performance of our SACFD baseline as for the suboptimally-trained model. For Can, it demonstrates similar training dynamics to the experiments for the suboptimally-trained model, regaining its initial success rate after around 350k steps. Our approach outperforms the VLA baseline after just 50k steps for the Can task, remaining at this level quite stably. For Square and Assembly, our approach stays at the level of the VLA baseline with minor deviations during the course of training.

After a qualitative analysis of the behaviour of policies from our approach during evaluation and training, we were able to extract that the key factor leading to

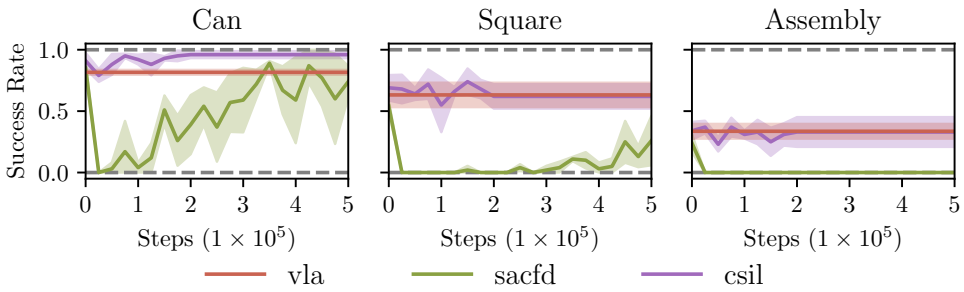


Figure 6.2.: Performance comparison in 3 robotic manipulation environments using a VLA fine-tuned for 30k steps. Success rates for our approach (green), SACFD (purple) and the VLA's base performance (red). Our approach surpasses the VLA base performance for Can and preserves it for the remaining 2 tasks. Uncertainty intervals depict one standard deviation.

failure in most unsuccessful episodes is the policy getting stuck close to completing either the task or a certain subtask. For the Square task, the policy consistently grasps the ring. However, the ring often gets stuck on the stick itself, and the policy is not able to resolve the issue. For Assembly, a surprising behaviour can be observed, where the policy is often close to grabbing the right object, yet either hovers around the object or gets stuck and does not try to regrasp but rather stays in its nearly optimal but still suboptimal state. This is surprising, as the negative reward should encourage the policy to complete the task as quickly as possible. Yet, the critic seems to encourage staying closer to a nearly complete subtask rather than moving on to the next subtask, where one risks failing to complete the next subtask, which might lead to a worse cumulative reward. Catastrophic failures, i.e. grasping the wrong object or missing the target object entirely, can also be seen for both Square and Assembly, though they are more prevalent for Assembly.

In summary, our results demonstrate that the SACFD baseline consistently exhibits a large initial performance drop, where the performance is either recovered after a substantial amount of steps or is not recovered at all. In contrast, our approach does not exhibit a large initial drop and improves on the suboptimally fine-tuned base model for both Can and Square. However, when evaluating our approach with a more optimally fine-tuned base model on the more complex Square and Assembly tasks, the performance remains close to the VLA baseline level, with

qualitative analysis suggesting a tendency to get stuck in near-completion states.

6.2. Ablating the Residual

This section investigates the impact of the architectural choice of fine-tuning a residual. For comparison, we evaluate the HETSTAT's performance under 2 configurations in addition to our evaluation of the residual for the π_0 model trained on a balanced dataset. First, by employing it directly for action projection. Second, by utilising an ensemble approach where the HETSTAT's projected action is averaged with the VLA's action to compute the final action executed in the environment.

Figure 6.3 shows that the residual matches or sometimes even outperforms the VLA base performance after pre-training. During the course of training, the residual improves on its initial performance for Can. For Square and Assembly, it does not significantly improve on the VLA baseline performance. In comparison, using the HETSTAT as an action head demonstrates inferior performance to the VLA baseline after pre-training, only surpassing the VLA baseline performance for the Can task. Using the HETSTAT as part of an ensemble leads to the initial performance being closer to the VLA baseline performance compared to using the HETSTAT actions alone. However, the ensemble method still stays below the baseline given by VLA during the entire course of training and does not meaningfully improve on its initial performance for all 3 environments.

In comparison to simply using the HETSTAT as an action head, using it as part of an ensemble likely leads to an improvement in initial performance as the suboptimal HETSTAT actions are averaged out with the better-fitted π_0 actions. Crucially, however, the ensemble limits the approach's potential as the actions are to 50% dependent on π_0's actions. Using the HETSTAT as a residual allows us to leverage the initial performance of the VLA while also not limiting the action space through combining the actions prior to projecting them into the action range.
In summary, using HETSTAT as a residual outperforms both the simple use of the HETSTAT policy as an action projection and also using it as part of an ensemble. Crucially, our approach is currently not able to outperform the baseline for the model fine-tuned using a balanced dataset outside of the Can task. The next section will explore what we believe to be the current main issue of our approach.

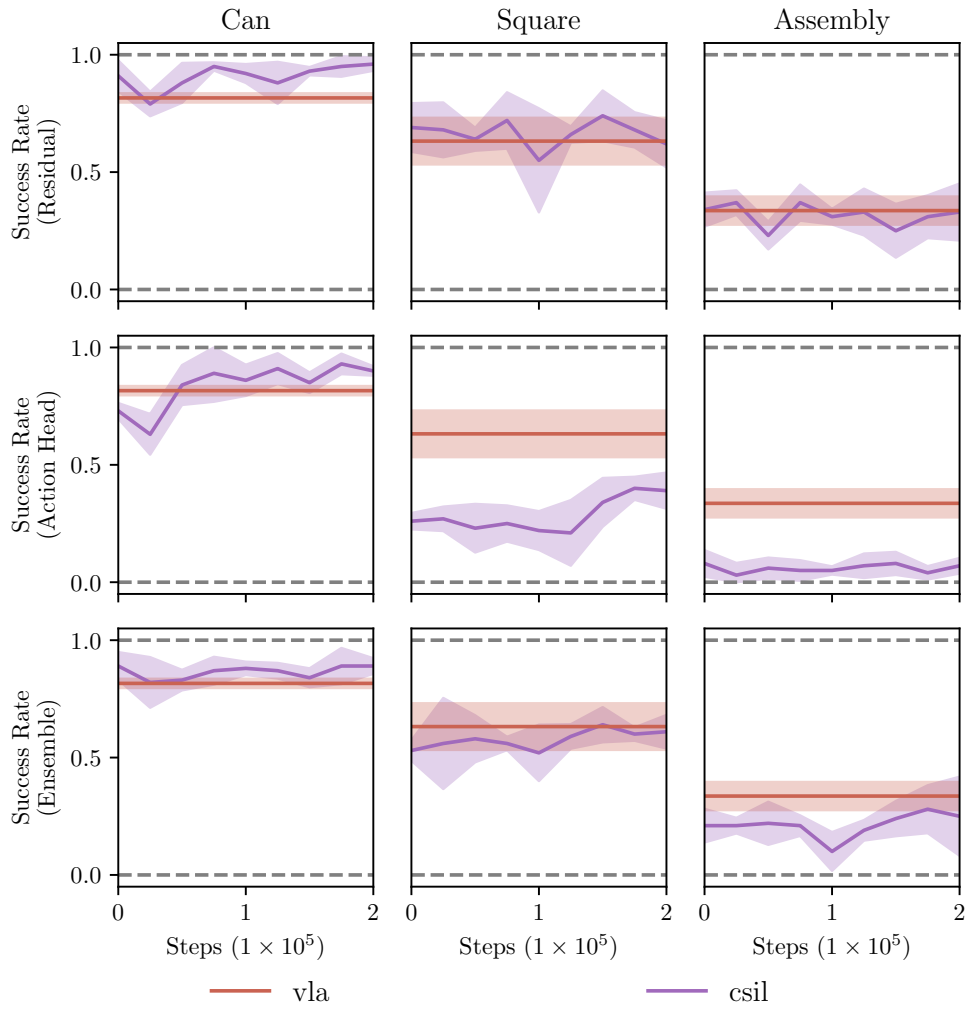


Figure 6.3.: Comparison of training a residual, using an ensemble approach and using the HETSTAT as the only action head for the π_0 model trained on a balanced dataset. Uncertainty intervals depict one standard deviation.

6.3. Investigating the Stationarity Issues

This section investigates an unexpected and yet crucial issue to be addressed: the loss of stationarity of the HETSTAT for high-dimensional inputs during pre-training. In particular, we observed the entropy of sampled actions for online states being indistinguishably close to the entropy of sampled actions for expert states.

6.3.1. Outlining the Problem

CSIL [38] relies strongly on the policy's ability for uncertainty quantification, achieved by predicting a larger variance and therefore a higher entropy of sampled actions for online states. This is apparent in the coherent reward, which the entire method revolves around. Additionally, the difference in entropy is important for improving our policy during RL, as we need to explore the state space, which is directly linked to the online entropy, as a higher online entropy leads to more random actions, leading to a more diverse set of states being visited. CSIL introduced the HETSTAT architecture to ensure that action distributions for states from online data have a high entropy even after pre-training. To investigate the non-separation issue of expert and online entropy, we evaluated the policy's online entropy every 10k steps during pre-training by running 25 episodes using stochastically sampled actions and reporting the mean online entropy. We ran the experiment for the high-dimensional VLA embedding case, where we additionally concatenated the joint states to the input. As a baseline, we ran the same experiment for the low-dimensional case, where we use the joint and object states as input for the HETSTAT. Figure 6.4 shows that the HETSTAT has a significantly higher online entropy than expert entropy in the low-dimensional input setting. Figure 6.4 also shows that the HETSTAT is unable to distinguish between online and expert states in the high-dimensional input setting, as the action entropies for online and expert states are almost identical. These results hint at a larger problem: π_0 may largely rely on the joint states to solve the tasks, and therefore, the embeddings for the online and expert states may be too similar.

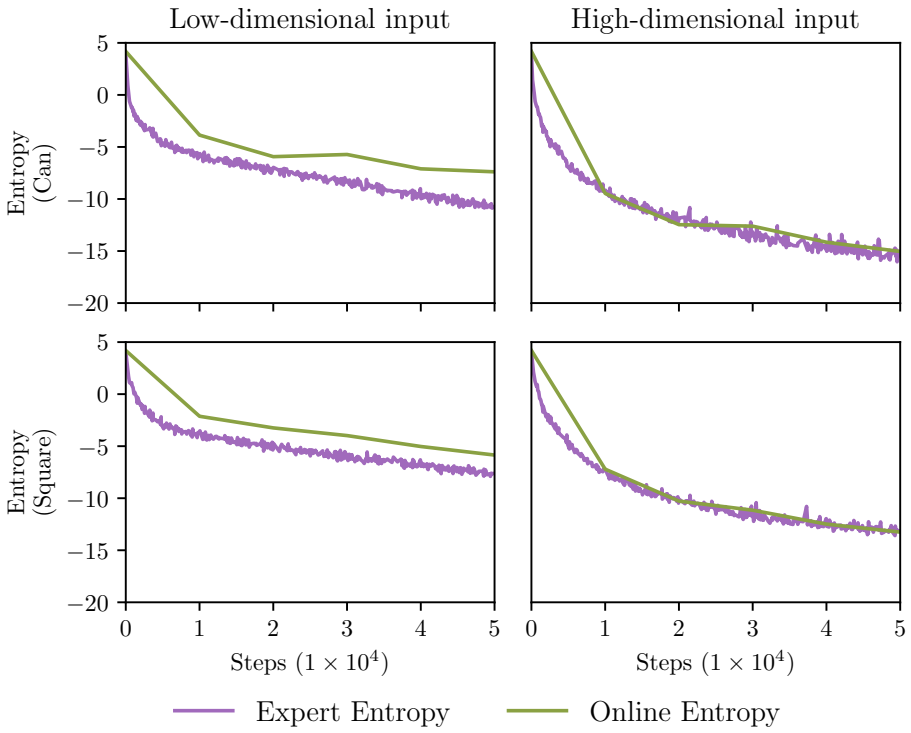


Figure 6.4.: Comparison of entropy of actions for online and expert states during the course of pre-training between the low-dimensional input setting and high-dimensional VLA input setting. For the low-dimensional case, the figure depicts the desired behaviour, where the online entropy is visibly higher than the expert entropy towards the end of training. For the high-dimensional case, the figure shows the online and expert entropy not separating during the course of training.

6.3.2. Investigating Embeddings

To isolate the cause of the issue, we repeated the experiments from Figure 6.4 for the high-dimensional setting, additionally concatenating oracle information to the inputs. In particular, we added the low-dimensional object state information. This should help the HETSTAT in distinguishing between online and expert states, as it does not have to rely on the embeddings. To investigate the effect of the embeddings on the problem, we additionally compare using only parts of the embeddings with using the whole embedding in combination with the low-dimensional state and object state information. Figure 6.5 shows that extending the input with the oracle information recovers the desired gap between the online and expert entropy. Figure 6.5 also shows that the gap decreases when enlarging the part of the embeddings used for the inputs. The original π_0 embeddings are 1024-dimensional.

6.3.3. Fully Fine-tuning π_0

So far, we have used π_0 models fine-tuned using LORA [20] as fully fine-tuning π_0 requires significantly more computational resources in comparison to fine-tuning π_0 with LORA. π_0's LORA training configuration freezes the vision-encoder however. To investigate whether this negatively impacts the quality of embeddings and therefore the HETSTAT's stationarity, we fully fine-tuned π_0 for 30k steps on the balanced dataset that includes 4 tasks. Fully fine-tuning refers to updating all weights of the network, including the vision encoders, as opposed to updating only parts of the network. We evaluated the model every 10k steps during pre-training for 25 episodes each, sampling the actions stochastically and logging the mean entropy of all actions. We compare the fully fine-tuned model against the model that was trained on the balanced dataset, where the vision encoder was frozen. Figure 6.6 shows that both models have similar pre-training dynamics. Only minor differences between the two models can be seen. While the online entropy stays almost exactly on the expert entropy during training for the model with a frozen vision encoder, the online entropy for the model where the vision encoder was updated during fine-tuning deviates slightly at 2 points during training. In our opinion, this deviation is negligible and does rectify the additional compute required for fully fine-tuning π_0. Rather, we believe there is another, larger cause leading to the stationarity issue that we have not identified yet, independent of whether the vision encoder was frozen or not during the fine-tuning.

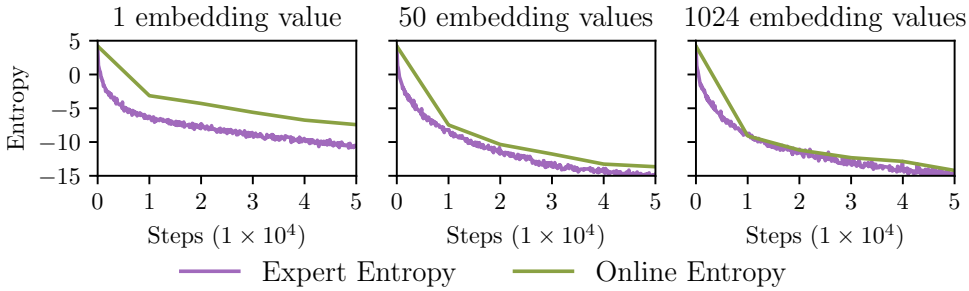


Figure 6.5.: Comparison of the policy's entropy for online and expert states during the course of pre-training when adding the low-dimensional object state information to the inputs and only using parts of π_0's embeddings. Most importantly, adding the object state information recovers the desired gap between online and expert entropy. Notably, the gap smallens when enlarging the part of the embeddings that is used as input. The original π_0 embeddings are 1024-dimensional.

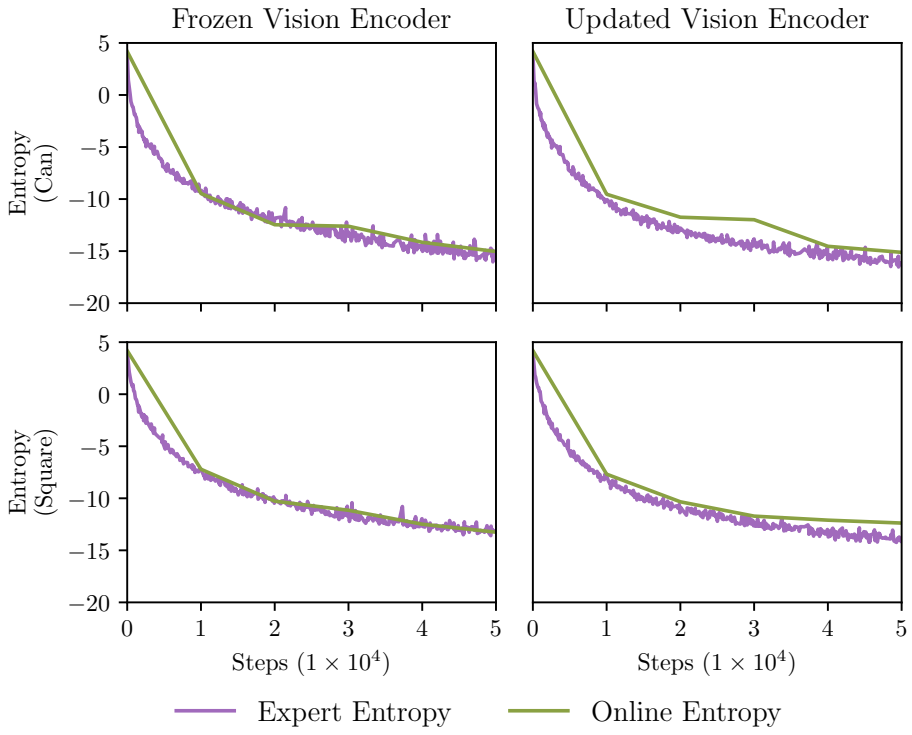


Figure 6.6.: Comparison of the policy's entropy for online and expert states during the course of pre-training between a π_0 model fine-tuned without updating the vision encoder and one with the vision encoder updated during training. There is no significant difference in their dynamics except for 2 small exceptions for the model with the updated vision encoder.

7. Conclusion

This chapter concludes the thesis by examining the limitations of the proposed approach, suggesting directions for future research, and summarising the key contributions. The integration of CSIL and residual training with VLA post-training shows potential for improving the performance of VLAs by increasing their robustness to out-of-distribution states. However, several challenges remain to be addressed.

7.1. Limitations

Our approach is limited in a variety of areas, warranting consideration. These limitations span from implementation challenges to real-world, practical challenges.

A core limitation of the approach in its current state is the problem regarding the non-stationarity of the HETSTAT for the high-dimensional π_0 embeddings as described in section 6.3. Solving this issue may lead to far superior results as both the reward signal and the exploration would benefit significantly.

The issue of non-stationarity of the HETSTAT highlights a second limitation of the approach: the requirement for a good initial policy. To be able to improve the robustness of a policy, it needs to be at a sufficiently high level to differentiate between online and expert entropy states and also to guide the exploration. At the current state of BC, this requires a sufficient amount of high-quality expert data, which may be cost- and labour-intensive to collect, especially for complex, long-horizon tasks.

From a practical perspective, the sample inefficiency of our approach needs to be addressed to enable real-world application. Training a policy with RL for 500 or more episodes in the real world is infeasible. Solving the issue of non-stationarity may increase the sample efficiency manyfold. Still, one needs to consider that a majority of the initial training may be attributed to the critic bootstrapping the state-action value until convergence. Yet, simply increasing the update-to-data

ratio of the critic to accelerate the bootstrapping currently leads to instabilities.

A second challenge of applying our approach in the real world is safety. As we evaluated our approach in simulation, we did not consider joint constraints or safe exploration. For real-world applicability, one needs to be able to assure the safety of the approach used, as it would otherwise endanger the robot and the humans around it. Reducing the labour needed for monitoring and managing the robot during its learning process may lead one to consider learning a second policy that resets the environment to a valid initial state. This approach would increase autonomy but also requires strong safety assurances to enable autonomous learning of the robot.

Finally, our current approach is intensive in hyperparameter tuning. The different components of BC pre-training and RL refinement each have their own set of parameters. Due to the computational complexity, especially of the RL refinement, the tuning of these parameters is very time-intensive.

7.2. Future Work

Based on the identified limitations, this section highlights promising directions for future research surrounding our approach.

A key direction is a further investigation on the problem of the degrading stationarity of the HETSTAT for high-dimensional VLA embeddings. Currently, it is unclear why the embeddings lead to a breakdown of the stationarity. A key aspect seems to be that π_0 may be overreliant on the movements of the end-effector itself and may be disregarding information about the object. Understanding the causes of the phenomenon and solving the issue would likely improve the quality of the reward signals significantly.

Addressing the challenges surrounding real-world applicability may be a further promising direction. As current VLAs are trained on large, diverse, real-world datasets, applying our approach in the real world is a natural step, as it would also address key concerns about the quality of embeddings produced by the VLA's vision-encoder, as they are largely not trained on simulation data. To close the gap to real-world evaluations, there are 2 major avenues for future research: safety and sample efficiency.

Safety is a natural direction, as the robots acting in the real world can be dangerous for humans and the environment around the setup. To address safety concerns, one could look at constraining the possible actions of the robot. An interesting idea could be introducing more advanced controllers that handle

collision and safety constraints. Another idea could be introducing inductive biases that transform the action space of the network to only output actions that are within safe bounds.

Increasing sample efficiency is an important direction of future research, as our current approach uses a few hundred episodes for training. Monitoring and supervising such training is very labour-, time-, and cost-intensive. Addressing the sample inefficiency of our current approach could also accelerate related research.

Finally, research on the pre-training of our policy is an intriguing direction for future research. There are 2 key aspects that come to mind: Reducing the amount of demonstration data needed for training a sufficiently proficient policy and enabling the use of suboptimal data for pre-training a policy. Both directions would lower the bar for training a policy to complete a certain task of interest. Enabling the use of suboptimal data would also allow us to use a larger portion of the available robotic data for training policies.

7.3. Summary

This thesis has presented an approach for post-training VLAs using IRL. The core of our method is the application of CSIL to VLAs using the introduction of a residual.

A key concept of CSIL is extracting a reward function from a policy that increases robustness by rewarding states that are in-distribution of our expert data. A key ingredient of our approach is combining a base model that guides exploration with a learnable residual that is not restricted by the base model.

Our experiments yielded mixed results. While the proposed method consistently matched the performance of the VLA, we demonstrated the capacity for improvement for certain environments. This finding underscores the potential of our approach to enhance the reliability VLAs on tasks they solve suboptimally, offering a practical method for fine-tuning that does not depend on handcrafted reward signals.

Crucially, this work identified a key technical challenge surrounding our approach: a degradation of the policy's uncertainty estimation when applied to the high-dimensional embeddings produced by VLAs. This limitation represents a primary bottleneck but also highlights a critical avenue for future research.

In conclusion, this thesis demonstrates parts of the potential of combining residual learning with IRL for the post-training of VLAs. While the full potential is not currently demonstrated, the method's performance for scenarios where the

main issue of non-stationarity is not present suggests potential for applications where the VLA is able to solve a task unreliably yet needs fine-tuning for a stable, robust execution of the desired behaviour.

7.4. Acknowledgements

I would like to thank Joe Watson for the insightful discussions during the thesis. I would also like to thank Daniel Palenicek and Theo Gruner for the supervision during the project. Additionally, I would like to thank Tim Schneider and Daniel Palenicek for maintaining the IAS cluster on which a majority of the experiments were run. I acknowledge the funding provided by the "Konrad Zuse School of Excellence in Learning and Intelligent Systems (ELIZA)" as part of a research-oriented master's scholarship.

Bibliography

[1] Lars Ankile et al. “From Imitation to Refinement - Residual Rl for Precise Assembly”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2024.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450.

[3] Lucas Beyer et al. *PaliGemma: A versatile 3B VLM for transfer*. 2024. arXiv: 2407.07726.

[4] Kevin Black et al. *π0: A Vision-Language-Action Flow Model for General Robot Control*. 2024. arXiv: 2410.24164.

[5] Nico Bohlinger and Klaus Dorer. “RL-X: A Deep Reinforcement Learning Library (not only) for RoboCup”. In: *Robot World Cup*. Springer, 2023.

[6] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. 2018. URL: http://github.com/jax-ml/jax.

[7] Anthony Brohan et al. *RT-1: Robotics Transformer for Real-World Control at Scale*. 2022. arXiv: 2212.06817.

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

[9] Mathilde Caron et al. *Emerging Properties in Self-Supervised Vision Transformers*. 2021. arXiv: 2104.14294.

[10] Xi Chen and Xiao Wang. “PaLI: Scaling Language-Image Learning in 100+ Languages”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[11] Yuhui Chen et al. *ConRFT: A Reinforced Fine-tuning Method for VLA Models via Consistency Policy*. 2025. arXiv: 2502.05450.

[12] Embodiment Collaboration et al. *Open X-Embodiment: Robotic Learning Datasets and RT-X Models*. 2025. arXiv: 2310.08864.

[13] David Cox and H. D. Miller. *The theory of stochastic processes*. 1965.

[14] DeepSeek-AI. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. 2025. arXiv: 2501.12948.

[15] Jonas Eschmann. "Reward function design in reinforcement learning". In: *Reinforcement learning algorithms: Analysis and Applications* (2021), pp. 25–33.

[16] Lasse Espeholt et al. "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures". In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2018.

[17] Hui Hwang Goh et al. "An assessment of multistage reward function design for deep reinforcement learning-based microgrid energy management". In: *IEEE Transactions on Smart Grid* 13.6 (2022), pp. 4300–4311.

[18] Aaron Grattafiori et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783.

[19] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: (2017).

[20] Edward Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *International Conference on Learning Representations (ICLR)*. 2022.

[21] Siddharth Karamcheti et al. "Prismatic VLMs: Investigating the Design Space of Visually-Conditioned Language Models". In: *International Conference on Machine Learning (ICML)*. 2024.

[22] Moo Jin Kim, Chelsea Finn, and Percy Liang. *Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success*. 2025. arXiv: 2502.19645.

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

[24] Yaron Lipman et al. "Flow Matching for Generative Modeling". In: *International Conference on Learning Representations (ICLR)*. 2023.

[25] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations (ICLR)*. 2017.

[26] Guanxing Lu et al. *VLA-RL: Towards Masterful and General Robotic Manipulation with Scalable Reinforcement Learning*. 2025. arXiv: 2505.18719.

[27] Ajay Mandlekar et al. "MimicGen: A Data Generation System for Scalable Robot Learning using Human Demonstrations". In: *Conference on Robot Learning (CoRL)*. 2023.

[28] Ajay Mandlekar et al. “What Matters in Learning from Offline Human Demonstrations for Robot Manipulation”. In: *Conference on Robot Learning (CoRL)*. 2021.

[29] Dicksiano C Melo, Marcos ROA Maximo, and Adilson Marques da Cunha. “Learning push recovery behaviors for humanoid walking using deep reinforcement learning”. In: *Journal of Intelligent & Robotic Systems* 106.1 (2022), p. 8.

[30] Lassi Meronen, Martin Trapp, and Arno Solin. “Periodic activation functions induce stationarity”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[31] Octo Model Team et al. “Octo: An Open-Source Generalist Robot Policy”. In: *Proceedings of Robotics: Science and Systems*. 2024.

[32] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: Advances in Neural Information Processing Systems (NeurIPS). 2022.

[33] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752.

[34] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning”. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[35] Manuel Sage, Martin Staniszewski, and Yaoyao Fiona Zhao. “Economic Battery Storage Dispatch with Deep Reinforcement Learning from Rule-Based Demonstrations”. In: *International Conference on Control, Automation and Diagnosis (ICCAD)*. 2023.

[36] John Schulman. *Approximating KL Divergence*. Online blog post. Accessed: July 15, 2025. 2020. URL: http://joschu.net/blog/kl-approx.html.

[37] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347.

[38] Joe Watson, Sandy H. Huang, and Nicolas Heess. “Coherent Soft Imitation Learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023.

[39] Wenke Xia et al. *Robotic Policy Learning via Human-assisted Action Preference Optimization*. 2025. arXiv: 2506.07127.

[40] Xiu Yuan et al. *Policy Decorator: Model-Agnostic Online Refinement for Large Policy Model*. 2024. arXiv: 2412.13630.

[41] Yuke Zhu et al. *robosuite: A Modular Simulation Framework and Benchmark for Robot Learning*. 2020. arXiv: 2009.12293.

A. Validation of Codebase

To verify the correctness of our reimplementation of the CSIL codebase, we run the same experiments on the Robosuite [41] environments as in Watson, Huang, and Heess (2023). Due to time constraints, we are only able to run each experiment with 1 seed. Figure A.1 and A.2 show that our implementation matches the original implementation for both the low-dimensional input setting as well as the image-based setting utilising the ResNet architecture [38, 16]. An important caveat is the increased noise in our results. This is due to our validation only running 1 seed per experiment.

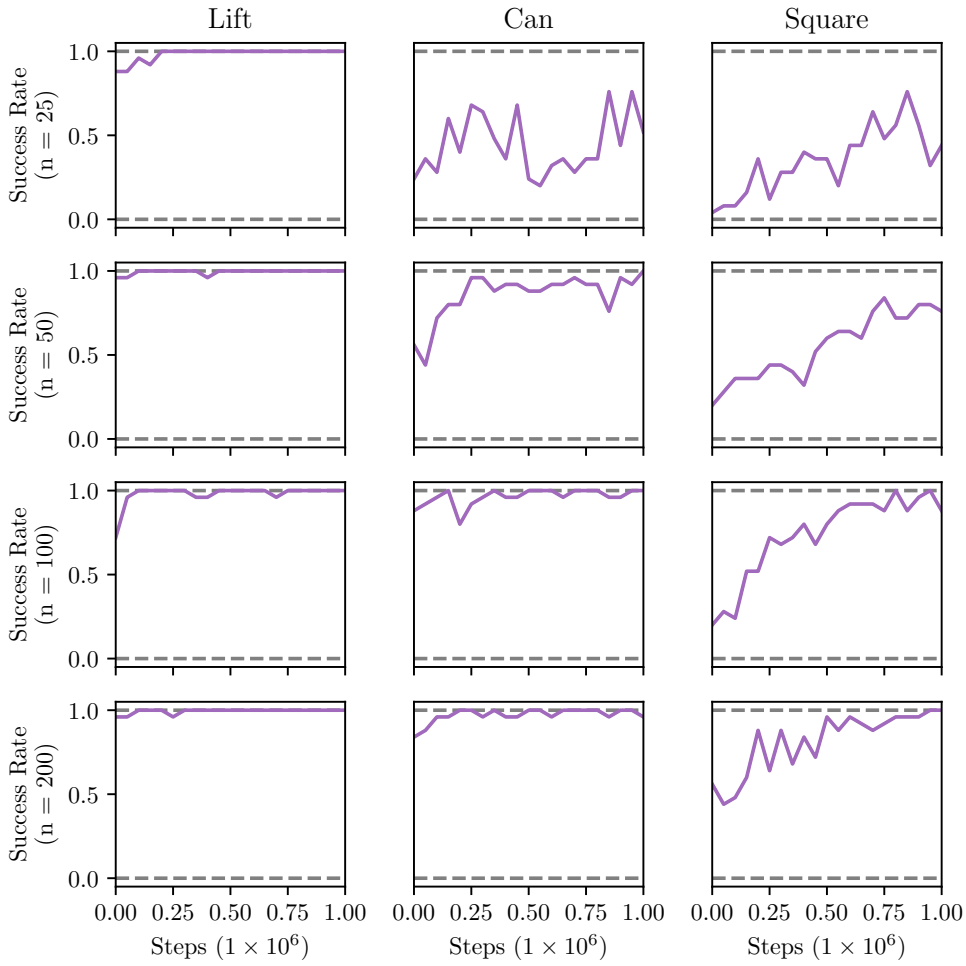


Figure A.1.: Results from our codebase for the low-dimensional state information inputs replicating the CSIL results [38]. Due to time constraints, we were only able to run 1 seed per experiment, while the original experiments were run with 10 seeds each. The number of demonstrations used for the expert dataset is specified in each row.

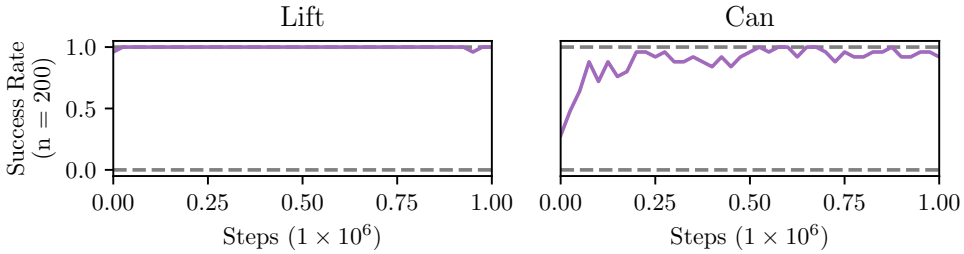


Figure A.2.: Results from our codebase for the ResNet-based replicating the CSIL results [38]. Due to time constraints, we were only able to run 1 seed per experiment. n specifies the number of demonstrations used for the expert dataset.

B. Experimental Details

We pre-train our residual policy for 5k steps using a learning rate of 1e-4. We initialise our critic for 20k steps with a learning rate of 1e-4, which is roughly until the mean state-action value converges. Both approaches use a batch size of 256, AdamW [25] as an optimiser, a stochastic actor during learning and a deterministic actor during evaluation. The critic, policy and coherent reward consist of 2 layers with 1024 hidden units, followed by a bottleneck layer with 48 hidden units using ELU activations. For all 3 networks, we employ layer normalisation [2] after the first layer of the MLP. The critic has a final layer with 1 hidden unit, while the policy and coherent reward have another layer with 1024 hidden units using the stationary activations. We use periodic ReLU activations as stationary activations. We refine our reward using a learning rate of 1e-3. We used negative CSIL rewards scaled by 0.16 to ensure they lie around $[-1, 0]$. For SACFD and our approach, we use a critic and policy learning rate of 3e-4.