SYNAPSE: SYmbolic Neural-Aided Preference Synthesis Engine

Sadanand Modak[1]*, Noah Patton[1], Isil Dillig[1], Joydeep Biswas[1]

[1]Department of Computer Science, The University of Texas at Austin
{smodak11, npatt, isil, joydeepb}@cs.utexas.edu

Abstract

This paper addresses the problem of *preference learning*, which aims to align robot behaviors through learning user-specific preferences (*e.g.* "good pull-over location") from visual demonstrations. Despite its similarity to learning *factual* concepts (*e.g.* "red door"), preference learning is a fundamentally harder problem due to its subjective nature and the paucity of person-specific training data. We address this problem using a novel framework called SYNAPSE, which is a neuro-symbolic approach designed to efficiently learn preferential concepts from limited data. SYNAPSE represents preferences as neuro-symbolic programs – facilitating inspection of individual parts for alignment – in a domain-specific language (DSL) that operates over images and leverages a novel combination of visual parsing, large language models, and program synthesis to learn programs representing individual preferences. We perform extensive evaluations on various preferential concepts as well as user case studies demonstrating its ability to align well with dissimilar user preferences. Our method significantly outperforms baselines, especially when it comes to out-of-distribution generalization. We show the importance of the design choices in the framework through multiple ablation studies. Code, additional results and supplementary material can be found on the website: `https://amrl.cs.utexas.edu/synapse`

1 Introduction

Imagine trying to come up with a definition of *"a good taxi drop-off location"*. One person may consider a spot to be a good drop-off location depending on whether it is close to the door of a building, while someone else might want it in the shade. Such concepts vary from person to person and inherently depend on their preferences. We call them *preferential concepts*, and we are interested in the problem of *preference learning* from visual input. Learning preferences is important because we want robots, and systems in general, that are customizable and can adapt to end-users (*e.g.* home robots). This problem is quite related to the task of visual concept learning, wherein much of the work focuses on learning concepts such as *having the color red* or *being to the left of another object* (Srivastava, Labutov, and Mitchell 2017; Mao et al. 2019; Han et al. 2019; Chen et al. 2021; Mei et al. 2022; Hsu, Mao, and Wu 2023; Wang et al. 2023; Hsu et al. 2023). All such prior work assumes there is a *ground-truth* for the concept, *i.e.* the definition of the concept does not differ among people, and as a consequence, sufficiently many examples are available, and can be objectively evaluated. We refer to such concepts as *factual concepts*. While most prior work that learns visual concepts exploits the availability of large datasets such as CLEVR (Johnson et al. 2017), those methods cannot be applied to preference learning because it is a data-impoverished setting by its very nature: a single individual can put up with providing only so much data! This limitation is also present in most of the preference learning work in the reinforcement learning literature as well, where human preferences are represented as neural networks or latent reward models (Ouyang et al. 2022; Busa-Fekete et al. 2013; Wirth, Fürnkranz, and Neumann 2016). Furthermore, because preferences are inherently individual, they can depend on entirely different (auxiliary lower-level) concepts, such as in the drop-off location example above (*i.e.* based on *proximity to door* as opposed to *being in the shade*). This requires learning novel visual concepts in a *hierarchical* manner, *i.e.* first learning the auxiliary concepts and then the ego concept. Lastly, coming up with a complete definition of a preferential concept at once is itself a hard problem: it is much easier for someone to *show* examples one-at-a-time that satisfy their intuition as humans tend to *build* their notion of a preferential concept over time. Thus, preference learning calls for an approach that can handle *incremental learning* from visual demonstrations, since we might not have access to a bunch of samples in one-go but would still like a good enough model of the preference based on the information at the time.

To address these challenges, we present SYNAPSE, a novel framework that learns human preferences in a data-efficient manner. In contrast to prior preference learning approaches which take in weak reward signals to learn preferences, we use a more direct form of a signal, which consists of a robot-demonstration and a natural language (NL) explanation for the preference. We use NL input to identify new concepts to be learned as well as how to compose them, thus representing the preference qualitatively. However, in addition to learning new concepts or composing existing ones, preferences also have a quantitative aspect. For instance, to be a good drop-off spot, it should be close to

*Corresponding author.

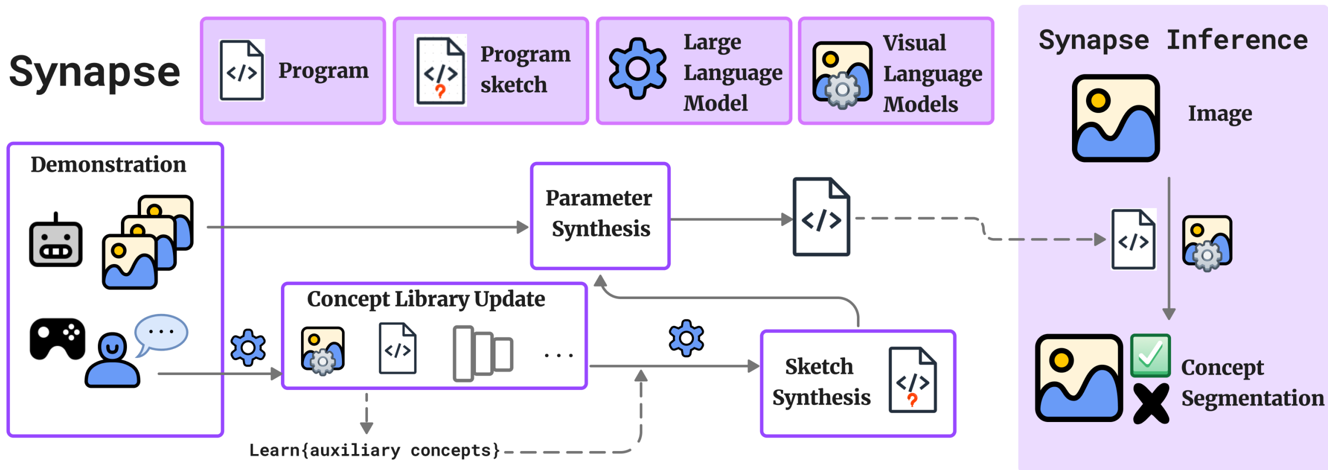

Figure 1: **Overview.** Human preferences have both *qualitative* and *quantitative* aspects. SYNAPSE first learns the necessary predicates (*a.k.a.* auxiliary concepts) needed to represent the preference from the NL input. It then synthesizes a program sketch which likely has some quantitative holes. This sketch represents the preference *qualitatively*. Finally, the holes are filled up by an optimization process that uses the physical demonstration data, thereby capturing the *quantitative* part of the preference.

a door, but exactly *how close* is a personal preference. This is where the demonstrations come into play and allow us to infer quantitative aspects of the preference that are hard to capture via natural language alone. Finally, to allow *incremental*, *sample-efficient* learning, SYNAPSE expresses preferential concepts as programs in a *neuro-symbolic* domain specific language (DSL) operating over images, and learns these programs based on demonstrations. Such a programmatic representation also facilitates *life-long learning*, allowing incremental changes to the learnt program as new demonstrations arrive.

Figure 1 shows a schematic of our proposed SYNAPSE framework. Given a user demonstration (*i.e.* the physical demonstration and NL input), the general workflow of SYNAPSE has three main components. First, SYNAPSE leverages the user's NL explanation, along with SYNAPSE's existing *concept library* (a collection of auxiliary learned concepts so far), to ground the *concepts* needed to represent the user's preference. If the NL explanation contains concepts that are not part of SYNAPSE's existing concept library, SYNAPSE may query the user for additional demonstrations of the auxiliary concept, which are then used to update SYNAPSE's concept library. Once the library contains all required concepts, SYNAPSE uses the NL explanation to generate a *program sketch* which is a program in our DSL with missing values (holes) for numeric parameters. Finally, SYNAPSE uses constrained optimization techniques (based on *maximum satisfiability* (Holtz, Guha, and Biswas 2021)) to find values of the numeric parameters that are maximally consistent with the user's physical demonstrations.

In what follows, we first describe the state-of-the-art in the field (Section 2) followed by details on the proposed framework (Section 3). This paper focuses mainly on mobility related preferences concerning navigation of robots and autonomous vehicles. However, to show that SYNAPSE works equally well for other concepts as well, we apply it to a well-studied preferential task in robot manipulation – tabletop object rearrangement (Ding et al. 2023). Section 4 presents our extensive evaluation on these domains, which includes user-studies as well as ablations. Finally, we conclude with a discussion on the opportunities for further work (Section 5). In summary, this paper contributes:

1. SYNAPSE, a neuro-symbolic framework to learn and evaluate preferences
2. a novel method for hierarchical lifelong learning from visual demonstrations and NL explanation
3. a comprehensive experimental evaluation of the proposed approach showing generalization to various domains

2 Related work

SYNAPSE positions itself in the larger field of concept learning and visual question answering (VQA). While there exist reinforcement learning based methods for preference learning, most of them fall in the imitation learning setting, where human preferences are represented via neural policies (Wilson, Fern, and Tadepalli 2012; Busa-Fekete et al. 2013) or latent reward models (Wirth, Fürnkranz, and Neumann 2016; Akrour et al. 2014; Christiano et al. 2017). Further, they do not deal with natural language, but rather take some form of weak preference signal as input. In the following discussion, we focus on work that is most closely-related to SYNAPSE.

Language Model Programs (LMPs). Generating executable programs from natural language is not a new idea. Many earlier works (Srivastava, Labutov, and Mitchell 2017; Yi et al. 2018; Mao et al. 2019; Chen et al. 2021; Mei et al. 2022) use custom semantic parsers to perform specific tasks. However, with the advent of Large Language Models (LLMs), LMPs have gained significant attention (Hu et al. 2023; Gupta and Kembhavi 2023; Surís, Menon, and Vondrick 2023; Liu et al. 2023a) due to the extensive knowledge that these foundation models possess. Code-as-Policies (Liang et al. 2023) pioneered the effort in this direction and demonstrated that LLMs can generate simple Python programs through recursive prompting for tasks ranging from drawing shapes to tabletop manipulation. While this can work for simpler and obvious tasks, it cannot be employed for learning preferences where people can associate different meanings to certain predicates, *e.g.* `is_close` might quantitatively differ between two individuals. As we describe later, SYNAPSE tries to remedy this by actively querying the human demonstrator for the auxiliary predicate meaning.

Neuro-symbolic concept learning. Neuro-symbolic approaches (Han et al. 2019; Mei et al. 2022; Kane et al. 2022; Silver et al. 2022; Hsu, Mao, and Wu 2023) couple the interpretability of rule-based symbolic AI with the strength of neural networks. One work (Srivastava, Labutov, and Mitchell 2017) uses a trained semantic parser to first extract useful feature definitions from a few statements describing the concept, which are then evaluated for each datapoint to build feature vectors on which standard classification can be done. NS-VQA (Yi et al. 2018) is another approach that uses a separately trained visual parser to generate a structured representation of objects in the image and a semantic parser trained to parse the question into a predefined DSL format, which is followed by Pythonic program execution. NS-CL (Mao et al. 2019) uses the same framework as NS-VQA, but instead of answering questions given the trained modules, it represents concepts as neural operators and tries to learn them given the question-answer pairs. However, most of these concept learning methods are data-hungry which makes them unfit for learning preferences.

Program synthesis. There is a rich literature on synthesizing programs from user-provided specifications in the programming languages community (Gulwani, Polozov, and Singh 2017; Patton et al. 2024; Gulwani 2011; Holtz, Guha, and Biswas 2021). There also has been work on synthesizing LTL formulas directly from natural language (Liu et al. 2023b, 2024). From the viewpoint of visual reasoning and concept learning, (Murali et al. 2019) tries to solve Visual Discrimination Puzzles (VDP) by synthesizing a discriminator expressed in first-order logic by performing a full-blown discrete search. However, this can quickly become inefficient as problem size scales. To tackle this, SYNAPSE uses natural language informed sketch generation and performs synthesis over the space of parameters.

3 Method

We define a *preference task* $\mathcal{T} := \langle O, Q, P \rangle$ as a tuple consisting of an observation space O, a query space Q, and a preference space P. A preference evaluator $\pi : O \times Q \to P$ accepts an observation and a query, and returns a preference value $p \in P$. The goal of *preference learning* is to synthesize a suitable evaluator π that accurately predicts a person's preferences. Since we focus on visual preferences in this paper, O is the space of RGB images, Q is pixel/location queries, and P is a segmentation mask over the image.

3.1 Representing Preferences

As stated earlier, a distinguishing feature of preference learning is that it has to be performed using *small* amounts of training data. To enable sample-efficient learning, SYNAPSE represents the preference evaluator π as a neuro-symbolic program in a DSL and synthesizes π from a small number of user demonstrations, where each demonstration includes a robot trajectory[1] (*i.e.* sequence of images) along with an NL explanation for the user's preference. The DSL, as shown in Figure 2, is parameterized over a *concept library* $\mathcal{C}$, which includes both boolean predicates $\mathcal{C}_b$ and non-boolean functions $\mathcal{C}_f$, built-in operators (*e.g.* $+, \leq, \ldots$), pre-trained neural models (*e.g.* zero-shot visual language models (VLMs)), as well as previously learned concepts and functions (expressed in the same DSL). At a high level, a program π in this DSL consists of (nested) if-then-else statements and is therefore conceptually similar to a decision tree. Each leaf of this decision tree is a preference (*e.g. good, neutral, bad*) drawn from the preference space P, which is assumed to be a finite set. Internal nodes of the decision tree are neuro-symbolic conditions ϕ, which include boolean combinations of predicates of the form $p_c(t_1, \ldots, t_n)$ where $p_c \in \mathcal{C}_b$ and each t_i is a neuro-symbolic term, that could be an operator (*e.g.* $\leq$), a neural module output, or a previously-learned concept (*e.g.* `is_close`).

3.2 Learning Preferences

Algorithm 1 summarizes the learning algorithm for synthesizing preference evaluator π from a set of demonstrations. As SYNAPSE is meant to be used in a lifelong learning setting, we present it as an incremental algorithm that takes one new demonstration d_{new} at each invocation and returns an updated preference evaluation function. As mentioned earlier, we represent each demonstration d as a pair (t, e) where t is a physical demonstration consisting of a sequence of images from a robot trajectory and e is the user's NL explanation for their preference. Given a demonstration d, we write $d.t$ and $d.e$ to denote its physical demonstration and NL component respectively. In addition to the new demonstration d_{new}, `Learn` takes three additional arguments, namely the previous set of demonstrations $\mathcal{D}$, the previously learned preference evaluator π_o (`None` for the first invocation), and the current concept library $\mathcal{C}$, which is initialized to contain only a set of built-in concepts (*i.e.* set of basic mathematical operations, camera homography, and pre-trained neural modules). `Learn` uses the old program π_o to bootstrap the learning process, and the previous demonstrations are required to ensure that the updated program is consistent with *all* demonstrations provided thus far. At a high level,

[1]this applies to the mobility concepts; for the tabletop rearrangement task it is just a single image demonstration

Inputs	Constants	Terms	Conditions	Programs
$q \in Q$ $o \in O$	$v \in$ {Int, Real, ...} $p \in P$	$t := q \mid o \mid v$ $\mid f(t_1, \ldots, t_n)$ where $f \in \mathcal{C}_f$	$\phi := p_c(t_1, \ldots, t_n)$ where $p_c \in \mathcal{C}_b$ $\mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi$	$\pi := p \mid$ `if` (ϕ) `then` π `else` π'

Figure 2: **SYNAPSE neuro-symbolic DSL.** Representing preference evaluator π parametrized over concept library $\mathcal{C}$

Algorithm 1: **The SYNAPSE learning framework**

Input: a set of previously seen demonstrations $\mathcal{D}$, the new demonstration d_{new}, previous sketch $\hat{\pi}_o$, and previous $\mathcal{C}$
Output: the new demonstrations set $\mathcal{D}'$, a neuro-symbolic preference evaluator π parameterized by new concept library $\mathcal{C}'$, new sketch $\hat{\pi}$ used to generate π

Learn$(\mathcal{D}, d_{new}, \hat{\pi}_o, \mathcal{C})$
    # Update concept library with new NL utterance
    $\mathcal{C}' \leftarrow$ UpdateConceptLibrary$(d_{new}.e, \mathcal{C})$
    # Get the updated sketch from the new demonstration and previous sketch
    $\hat{\pi} \leftarrow$ SketchSynth$(d_{new}, \hat{\pi}_o, \mathcal{C}')$
    # Fill the holes in $\hat{\pi}$ based on demonstrations $\mathcal{D}'$
    $\pi \leftarrow$ ParamSynth$(\hat{\pi}, \mathcal{D} \cup \{d_{new}\})$
    return $\mathcal{D} \cup \{d_{new}\}, \pi, \mathcal{C}', \hat{\pi}$

Algorithm 2: **Concept library update**

Input: a new NL explanation e and the previous concept library $\mathcal{C}$
Output: a new concept library $\mathcal{C}'$

1: UpdateConceptLibrary$(e, \mathcal{C})$
2:     $\mathcal{C}' \leftarrow \mathcal{C}$ *# Initialize new concept library*
3:     *# Get new visual groundings from NL and add to $\mathcal{C}'$*
4:     $g \leftarrow$ ExtractEntities$(e, \mathcal{C})$
5:     $\mathcal{C}' \leftarrow \mathcal{C}' \cup g$
6:     *# Extract new predicates from e*
7:     preds $\leftarrow$ ExtractPredicates$(e, \mathcal{C})$
8:     *# Recursively update concepts with user feedback*
9:     **for** pred $\in$ preds **where** pred $\notin \mathcal{C}$
10:         $\mathcal{D} \leftarrow \emptyset$ *# Empty initial demonstration set*
11:         $\mathcal{C}'' \leftarrow \mathcal{C}'$
12:         $\hat{\pi} \leftarrow$ None
13:         **do**
14:             $d \leftarrow$ QueryUserForDemonstration(pred)
15:             $\mathcal{D}, \pi, \mathcal{C}'', \hat{\pi} \leftarrow$ Learn$(\mathcal{D}, d, \hat{\pi}, \mathcal{C}'')$
16:         **while** d
17:         $\mathcal{C}' \leftarrow \mathcal{C}''$
18:     **return** $\mathcal{C}'$

the learning procedure consists of three steps which will be explained in the remainder of this section.

Concept Library Update. First, we check whether the existing concept library $\mathcal{C}$ is sufficient for successfully learning the desired preference evaluation function by analyzing the user's NL explanation e to extract concepts of interest. We differentiate between two types of concepts: (1) entities (*e.g.* car, door, sidewalk) and (2) predicates (*e.g.* far, near). Because we use an open-vocabulary VLM to find entities of interest in the current observation, new entity concepts do not require interacting with the user. On the other hand, if NL contains new predicates that are not part of the existing concept library, it needs to query the user to provide suitable demonstrations. For instance, if user provides NL as *"this location is good because it is on the sidewalk, far from the person and the car, and not in the way"*, we would extract entities as *'sidewalk'*, *'person'*, and *'car'*, and auxiliary concepts/predicates as *'is_on'*, *'is_far'*, and *'in_way'* and if any of the predicates is not present in the current $\mathcal{C}$, we'll first learn it (*i.e.* applying SYNAPSE recursively for hierarchical learning) by querying the user for a few demonstrations.

Algorithm 2 summarizes this discussion. In lines 3--5, the `ExtractEntities` procedure uses an LLM to ground the entities used in the NL description and cross-reference them against existing entities in the concept library. Any new entities are added to the concept library without requiring user interaction, as we assume that any entity can be extracted from the observation using an open-set VLM. Lines 6-17, on the other hand, extract new *predicates* (auxiliary concepts) from the NL description and add it to the concept library. Since the semantics of these predicates cannot be assumed to be known a priori (unless they are already in the concept library), we must actively query the user to learn their semantics. Thus, the `QueryUserForDemonstration` procedure obtains new demonstrations, which are then used to synthesize these new predicates through recursive invocation of `Learn` at line 15, so that when the `UpdateConceptLibrary` procedure terminates, the new concept library $\mathcal{C}'$ contains all entities and predicates of interest.

Program Sketch Synthesis. Once SYNAPSE has all the required concepts as part of its library, it uses an LLM to synthesize a *program sketch*, which is a program with missing constants to be learned. We differentiate between program sketches and complete programs because the user's NL explanation is often sufficient to understand the general structure of the preference evaluation function but not its numeric parameters, which can only be accurately learned from the physical demonstrations. In particular, it first prompts the LLM to translate the NL explanation e to a pair (Φ, r) where Φ is a formula in conjunctive normal form (CNF) over the predicates in the concept library and r is the user's preference. Then, in a second step, SYNAPSE prompts the LLM to update the previous sketch $\hat{\pi}$ to a new one $\hat{\pi}'$ such that $\hat{\pi}'$ returns r when Φ evaluates to `True`. We found this two-stage process of first converting the NL explanation to a CNF formula and then prompting the LLM to

Algorithm 3: **Parameter synthesis**

Input: a program sketch $\hat{\pi}$, a set of demonstrations $\mathcal{D}$
Output: a complete program π

```
1:  ParamSynth(π̂, 𝒟)
2:    φ ← true # Initialize
3:    for d ∈ D
4:      # Perform partial evaluation on the sketch
         and demonstration to get a simplified
         sketch π̂′ and expected result r
5:      (π̂′, r) ← PartialEval(π̂, d)
6:      # Merge with condition
7:      φ ← φ ∧ ⟦π̂′⟧^r
8:      # Include negation for each parameter ∈ P
9:      for i ∈ P
10:       if (i ≠ r) φ ← φ ∧¬⟦π̂′⟧^i
11:   # Use solver to fill holes over symbolic features
12:   π ← Solver(φ)
13:   return π
```

repair the old sketch to work better in practice compared to prompting the LLM directly with all inputs (see Section 4). For our running example, the Φ would be *is_on('sidewalk') and is_far('person') and is_far('car') and not in_way()*, and r would be *'good'*.

Parameter Synthesis. As mentioned earlier, a program sketch contains unknown numeric parameters that arise from the ambiguity of NL, *e.g.* what does *"close"* mean in terms of distances between objects? Thus, the last step of the SYNAPSE pipeline utilizes the user's physical demonstrations to synthesize numeric parameters in the sketch. As summarized in Algorithm 3, it constructs a logical formula φ consistent with all demonstrations as follows: first, for each physical demonstration d, it partially evaluates $\hat{\pi}$ (line 5) by fully evaluating all expressions without any unknowns, yielding a much simpler sketch containing only unknowns to be synthesized but no other variables. For instance, if the sketch contains the predicate `distanceTo(car)`, we can use the observation from d to compute the actual distance between the location and the car. Next, let $[\![\hat{\pi}]\!]^i$ denote the condition under which $\hat{\pi}$ returns preference $i \in P$, and suppose that the current demonstration d illustrates preference class r. Since we would like the synthesized program to return r for demonstration d, $[\![\hat{\pi}]\!]^r$ should evaluate to `True`, while for all other preference classes i where $i \neq r$, $[\![\hat{\pi}]\!]^i$ should evaluate to `False`. Thus, the loop in lines 8-10 iteratively strengthens formula φ by conjoining it with $[\![\hat{\pi}]\!]^r$ and the negation of $[\![\hat{\pi}]\!]^i$ for any i distinct from r. Finally, we use an off-the-shelf constraint solver to obtain a model of the resulting formula[2] that is maximally consistent with the user's demonstrations. This results in a fully learned program that represents the user's preference.

[2]In general, the demonstrations may be noisy (*i.e.* φ could be unsatisfiable), which is quite often the case with real-world data. Thus, we use a MaxSMT solver (Bjørner, Phan, and Fleckenstein 2015) to maximize the number of satisfied clauses.

4 Evaluation

We first describe the experimental setup and the benchmark for mobility tasks, and then present the performance of SYNAPSE across four dimensions:

Q1. How does it compare against the state-of-the-art?
Q2. Can it easily and effectively extend to other domains?
Q3. Can it align well to dissimilar multi-user preferences?
Q4. How important are the various design choices?

Experimental Setup. We evaluate on three mobility-related preferential concepts: a) `CONTINGENCY`: *What is a good spot for a robot to pull over to in case of an emergency?*, b) `DROPOFF`: *What is a good location for an autonomous taxi to stop and drop-off a customer?*, and c) `PARKING`: *What is a good location for parking an autonomous car?*. In this work, we consider the preference space as binary only. The human demonstrations include the robot trajectories of the user driving the robot to the preferred location using a joystick, and NL description to explain the rationale for choosing that location. We use Grounded-SAM (Ren et al. 2024) zero-shot VLM for object detection, Depth Anything (Yang et al. 2024) for zero-shot depth estimation, and a custom terrain model with SegFormer architecture (Xie et al. 2021) finetuned on custom data, since we observed that open-set visual models perform pretty poor on terrain segmentation. We use these models to get segmentation masks of the neural concepts (*i.e.* objects and terrains) in the concept library. We use GPT-4 (Achiam et al. 2023) as the LLM for sketch synthesis. Lastly, as mentioned earlier, SYNAPSE can interactively query the user to clarify new concepts that are present in the user's NL explanation but not in the current concept library. In principle, SYNAPSE can query the user for both physical demonstrations and NL explanations. However, to reduce the burden on the user, SYNAPSE, by default, only queries the user for NL explanations of auxiliary concepts and performs synthesis of auxiliary concepts using NL explanations alone.

Baselines. We create a dataset of 815 labeled images taken from the UT Austin campus area, where the labels mark the locations on the images that are consistent with the intended user preference for each of the mobility tasks. We split the dataset into three sets: train, in-distribution test, and out-of-distribution test sets. The train and in-distribution sets belong to the same geographical region, while the out-of-distribution set belongs to a different region. Table 1 shows the comparison against various baselines (we only show the strongest variant of each baseline here). We use mean Intersection-Over-Union (mIOU) as the metric and evaluate the following baselines: (1) pure neural models based on SegFormer (SF) (Xie et al. 2021) architecture (with and without depth input) and DinoV2 (Oquab et al. 2023), both with pretrained weights and fine-tuned on our custom dataset; (2) NS-CL (Mao et al. 2019), a neurosymbolic concept learning approach for predominantly *factual* concepts, trained on our dataset; (3) `VisProg` (Gupta and Kembhavi 2023) which is a state-of-the-art VQA neurosymbolic method; and (4) `GPT4` (Achiam et al. 2023) vision. We find that SYNAPSE outperforms all baselines and improves

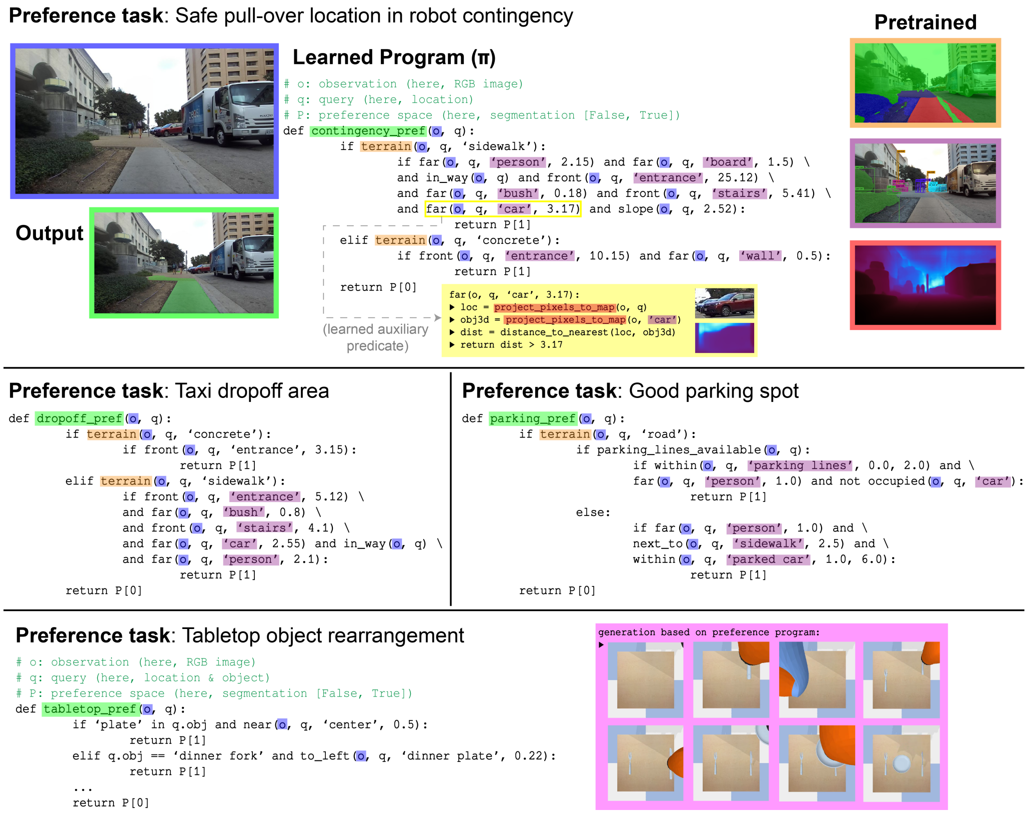

Figure 3: **Preference tasks.** We show evaluation on three mobility tasks and one manipulation task. SYNAPSE utilizes pre-trained module outputs and executes the learned program.

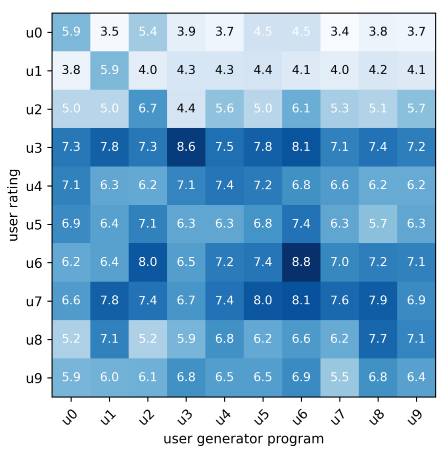

Figure 4: **User-study.** Higher entries around diagonal show good alignment between learned program and preference.

on the closest baseline by a significant margin on out-of-distribution test data – 74.07 *vs*. 57.42 for CONTINGENCY, 80.72 *vs*. 63.99 for DROPOFF, and 62.76 *vs*. 52.91 for PARKING. Further, even though SYNAPSE is trained on an order of magnitude fewer samples (for instance, 29 demonstrations for CONTINGENCY) than neural baselines (for instance, 224 images for CONTINGENCY), it matches or improves the baseline.

Generalize to other domains. We evaluate SYNAPSE on the tabletop object arrangement task: *Given a set of objects on dinner table, what is a good arrangement?*, as introduced in LLM-GROP (Ding et al. 2023), to show extension to other domains such as robot manipulation. Similar to the LLM-GROP work, we use 10 participants and utilise user ratings as the metric. This task consists of 8 sub-tasks with different sets of objects. We use five of these as our train tasks and the rest three as the test tasks. We collect one demonstration per train task from each user, where again the demonstration consists of the user showing the preferred object arrangement as well as a NL description. We use the baselines from the original paper. Table 3 summarizes the results where SYNAPSE outperforms the closest baseline by an average of 2.7 points across all tasks.

Multi-user preferences. For the LLM-GROP task, since we have learned preference programs for all 10 participants, we test the alignment of the learned programs with the different user preferences. For this, we generate object arrangements using learned programs for each user and then ask all other users to rate the arrangement. The results are summarized in Figure 4 which shows the average rating across the eight sub-tasks. For each user, the highest performance is attained by the program that was learned from the same user's demonstrations, which indicates good alignment.

Method / Split	CONTINGENCY			DROPOFF			PARKING		
	train	in-test	out-test	train	in-test	out-test	train	in-test	out-test
SYNAPSE	**77.64**	**76.29**	**74.07**	**79.32**	**80.18**	**80.72**	68.60	**66.87**	**62.76**
SF-RGBD-b5	76.48	67.81	56.11	77.69	70.70	52.39	**71.06**	65.72	49.99
DinoV2-g	73.65	60.93	51.23	79.50	72.17	59.10	67.06	62.04	52.78
NS-CL	69.76	69.63	63.65	71.26	70.38	63.99	46.92	43.71	45.23
VisProg	38.94	39.21	41.83	39.17	39.44	43.14	38.88	39.64	38.99
GPT4V	28.73	28.96	33.92	39.38	38.34	39.14	41.38	42.20	39.77

Table 1: Mean IOU (%) $\uparrow$ results for the three concepts. The train set represents the full set – SYNAPSE only needs 29 demonstrations (from the train area), while other fine-tuned (SegFormer, DinoV2) or trained (NS-CL) baselines use the full set.

Method / Feature	feat1	feat2	LLM	VLM	mIOU (%)
SYNAPSE	✓	✓	GPT-4 (Achiam et al. 2023)	DINO-SAM (Ren et al. 2024)	**76.11**
SYNAPSE-SynthDirect	✗	✗	GPT-4 (Achiam et al. 2023)	DINO-SAM (Ren et al. 2024)	60.74
SYNAPSE-SynthCaP	✗	✓	GPT-4 (Achiam et al. 2023)	DINO-SAM (Ren et al. 2024)	64.11
SYNAPSE-PaLM2	✓	✓	PaLM2 (Anil et al. 2023)	DINO-SAM (Ren et al. 2024)	71.62
SYNAPSE-GroupViT	✓	✓	GPT-4 (Achiam et al. 2023)	GroupViT (Xu et al. 2022)	73.41
SF-RGBD-b5	-	-	-	-	53.81
DinoV2-g	-	-	-	-	65.71

Table 2: The results for the ablation studies. Evaluation is on the full `CONTINGENCY` dataset.

Task #ID	1	2	3	4	5	6	7	8
SYNAPSE	**7.13**	**6.57**	**5.67**	**7.63**	**7.23**	**6.73**	**8.30**	**6.50**
LLM-GROP	4.07	3.27	3.37	5.83	4.40	4.50	5.80	2.70
LATP	3.93	1.70	2.60	2.10	3.23	2.93	1.93	2.33
GROP	2.60	2.47	2.87	2.37	2.37	2.77	3.57	2.33
TPRA	2.77	2.27	2.47	2.17	2.87	2.17	2.13	2.07

Table 3: User ratings on LLM-GROP (Ding et al. 2023) tabletop object rearrangement task on a scale of 1-10.

Ablations. We investigate four classes of ablations: (1) *NN-ablations*, in which we compare the performance of neural baselines (SF and DinoV2) against SYNAPSE when trained on the *same* number of samples (*i.e*. 29); (2) *LLM-based* in which we replace GPT-4 with different models in program synthesis part of the framework; (3) *VLM-based* ablations, where we test different VLMs for object detection in our framework; and (4) *framework* ablations where we test the following framework features: (a) *feat1:* whether it queries the user for auxiliary concepts, (b) *feat2:* whether it performs lifelong learning by building on its concept library. It can be seen from Table 2 that the NN-ablations perform poorly since they are exposed to so few training samples that they aren't able to generalize well to the full dataset. Changing the program synthesis process of SYNAPSE (*i.e*. not maintaining the library) or the LLM/VLM which in turn affects the accuracy of the program sketch and the parameters being synthesized, respectively, also has a significant impact on the performance.

More details on the evaluation and additional experiments are provided in the supplementary material.

5 Conclusion, Limitations & Future work

We presented SYNAPSE, a data-efficient, neuro-symbolic framework for learning preferential concepts from a small number of human demonstrations. We experimentally showed that SYNAPSE achieves strong generalization on new data and it outperforms the baselines by a large margin ($\approx 15\%$ mIOU). Further, we showed that SYNAPSE is able to align well with multi-user preferences. Finally, we also showed that SYNAPSE extends to other domains effectively. However, SYNAPSE has some potential limitations as well. First, SYNAPSE relies substantially on the quality of underlying neural modules and their capabilities. In our experiments, we observe that a careful selection of parameters and clever prompting is needed to achieve best performance. Further, SYNAPSE relies on the quality of the user's NL utterance as well as physical demonstrations for accurate synthesis. In practice, the demonstrations could be noisy and imperfect. Although SYNAPSE tries to compensate for slight inconsistencies in the user demonstration by using MaxSMT, however, to truly tackle this noise, a probabilistic approach to neurosymbolic programs needs to be explored. Finally, SYNAPSE as presented here doesn't take into account dynamically varying preferences, *i.e.* if a person's preference changes drastically between 1st and the nth sample, SYNAPSE would still give equal weight to both. A recency weighting approach might resolve this limitation.

Acknowledgements

This work is partially supported by the National Science Foundation (OIA-2219236, DGE-2125858, CCF-2319471, CCF-1918889, CCF-1901376). Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

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

Akrour, R.; Schoenauer, M.; Sebag, M.; and Souplet, J.-C. 2014. Programming by feedback. In *International Conference on Machine Learning*, 32, 1503–1511. JMLR. org.

Anil, R.; Dai, A. M.; Firat, O.; et al. 2023. PaLM 2 Technical Report. arXiv:2305.10403.

Bjørner, N.; Phan, A.-D.; and Fleckenstein, L. 2015. νz-an optimizing SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems: 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings 21*, 194–199. Springer.

Busa-Fekete, R.; Szörényi, B.; Weng, P.; Cheng, W.; and Hüllermeier, E. 2013. Preference-based evolutionary direct policy search. In *ICRA Workshop on autonomous learning*, volume 2.

Chen, Z.; Mao, J.; Wu, J.; Wong, K.-Y. K.; Tenenbaum, J. B.; and Gan, C. 2021. Grounding physical concepts of objects and events through dynamic visual reasoning. *arXiv preprint arXiv:2103.16564*.

Christiano, P. F.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; and Amodei, D. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Ding, Y.; Zhang, X.; Paxton, C.; and Zhang, S. 2023. Task and motion planning with large language models for object rearrangement. *arXiv preprint arXiv:2303.06247*.

Gulwani, S. 2011. Automating string processing in spreadsheets using input-output examples. In *Proc. of POPL*, 317–330.

Gulwani, S.; Polozov, O.; and Singh, R. 2017. Program Synthesis. volume 4, 1–119.

Gupta, T.; and Kembhavi, A. 2023. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14953–14962.

Han, C.; Mao, J.; Gan, C.; Tenenbaum, J.; and Wu, J. 2019. Visual concept-metaconcept learning. *Advances in Neural Information Processing Systems*, 32.

Holtz, J.; Guha, A.; and Biswas, J. 2021. Robot action selection learning via layered dimension informed program synthesis. In *Conference on Robot Learning*, 1471–1480. PMLR.

Hsu, J.; Mao, J.; Tenenbaum, J. B.; and Wu, J. 2023. What's Left? Concept Grounding with Logic-Enhanced Foundation Models. *arXiv preprint arXiv:2310.16035*.

Hsu, J.; Mao, J.; and Wu, J. 2023. Ns3d: Neuro-symbolic grounding of 3d objects and relations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2614–2623.

Hu, Z.; Lucchetti, F.; Schlesinger, C.; Saxena, Y.; Freeman, A.; Modak, S.; Guha, A.; and Biswas, J. 2023. Deploying and Evaluating LLMs to Program Service Mobile Robots. *arXiv preprint arXiv:2311.11183*.

Johnson, J.; Hariharan, B.; Van Der Maaten, L.; Fei-Fei, L.; Lawrence Zitnick, C.; and Girshick, R. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2901–2910.

Kane, B.; Gervits, F.; Scheutz, M.; and Marge, M. 2022. A System For Robot Concept Learning Through Situated Dialogue. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 659–662.

Liang, J.; Huang, W.; Xia, F.; Xu, P.; Hausman, K.; Ichter, B.; Florence, P.; and Zeng, A. 2023. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 9493–9500. IEEE.

Liu, B.; Jiang, Y.; Zhang, X.; Liu, Q.; Zhang, S.; Biswas, J.; and Stone, P. 2023a. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Liu, J. X.; Shah, A.; Rosen, E.; Jia, M.; Konidaris, G.; and Tellex, S. 2024. Skill Transfer for Temporal Task Specification. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2535–2541. IEEE.

Liu, J. X.; Yang, Z.; Idrees, I.; Liang, S.; Schornstein, B.; Tellex, S.; and Shah, A. 2023b. Grounding complex natural language commands for temporal tasks in unseen environments. In *Conference on Robot Learning*, 1084–1110. PMLR.

Mao, J.; Gan, C.; Kohli, P.; Tenenbaum, J. B.; and Wu, J. 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*.

Mei, L.; Mao, J.; Wang, Z.; Gan, C.; and Tenenbaum, J. B. 2022. FALCON: fast visual concept learning by integrating images, linguistic descriptions, and conceptual relations. *arXiv preprint arXiv:2203.16639*.

Murali, A.; Sehgal, A.; Krogmeier, P.; and Madhusudan, P. 2019. Composing Neural Learning and Symbolic Reasoning with an Application to Visual Discrimination. *arXiv preprint arXiv:1907.05878*.

Oquab, M.; Darcet, T.; Moutakanni, T.; Vo, H. V.; Szafraniec, M.; Khalidov, V.; Fernandez, P.; Haziza, D.; Massa, F.; El-Nouby, A.; Howes, R.; Huang, P.-Y.; Xu, H.; Sharma, V.; Li, S.-W.; Galuba, W.; Rabbat, M.; Assran, M.; Ballas, N.; Synnaeve, G.; Misra, I.; Jegou, H.; Mairal, J.;

Labatut, P.; Joulin, A.; and Bojanowski, P. 2023. DINOv2: Learning Robust Visual Features without Supervision.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744.

Patton, N.; Rahmani, K.; Missula, M.; Biswas, J.; and Dillig, I. 2024. Programming-by-Demonstration for Long-Horizon Robot Tasks. *Proc. ACM Program. Lang.*, 8(POPL).

Ren, T.; Liu, S.; Zeng, A.; Lin, J.; Li, K.; Cao, H.; Chen, J.; Huang, X.; Chen, Y.; Yan, F.; Zeng, Z.; Zhang, H.; Li, F.; Yang, J.; Li, H.; Jiang, Q.; and Zhang, L. 2024. Grounded SAM: Assembling Open-World Models for Diverse Visual Tasks. arXiv:2401.14159.

Silver, T.; Athalye, A.; Tenenbaum, J. B.; Lozano-Perez, T.; and Kaelbling, L. P. 2022. Learning neuro-symbolic skills for bilevel planning. *arXiv preprint arXiv:2206.10680*.

Srivastava, S.; Labutov, I.; and Mitchell, T. 2017. Joint concept learning and semantic parsing from natural language explanations. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, 1527–1536.

Surís, D.; Menon, S.; and Vondrick, C. 2023. Vipergpt: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128*.

Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; and Anandkumar, A. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Wilson, A.; Fern, A.; and Tadepalli, P. 2012. A bayesian approach for policy learning from trajectory preference queries. *Advances in neural information processing systems*, 25.

Wirth, C.; Fürnkranz, J.; and Neumann, G. 2016. Model-free preference-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Xie, E.; Wang, W.; Yu, Z.; Anandkumar, A.; Alvarez, J. M.; and Luo, P. 2021. SegFormer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34: 12077–12090.

Xu, J.; De Mello, S.; Liu, S.; Byeon, W.; Breuel, T.; Kautz, J.; and Wang, X. 2022. Groupvit: Semantic segmentation emerges from text supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18134–18144.

Yang, L.; Kang, B.; Huang, Z.; Xu, X.; Feng, J.; and Zhao, H. 2024. Depth anything: Unleashing the power of large-scale unlabeled data. *arXiv preprint arXiv:2401.10891*.

Yi, K.; Wu, J.; Gan, C.; Torralba, A.; Kohli, P.; and Tenenbaum, J. 2018. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, 31.