How will we evaluate?

Each team can submit one demonstration video of **less than 1 minute to showcase their project**. Upload your video to the Hugging Face Dataset in the LeRobot Worldwide Hackathon community. Each team must also submit the datasets recorded during the hackathon (if relevant). Instructions are provided on HF Org!

Hugging Face judges will select **30 projects** based on the teams' demonstration videos. The selection will consider the following criteria :

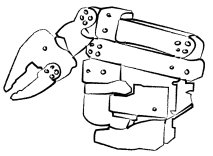
Creativity, Technical difficulty, Usability, and Real-world Impact

On June 16, the 30 demos will **be shared on Discord for community voting!**
Everyone can cast their vote in a poll: just one vote per person, so choose your favorite team wisely.

The teams with the most votes will win!

🥇 1st to 3rd teams: 1 Hope Jr Arm & 1 LeKiwi per team
🥈 4th and 5th teams: 1 Hope Jr Arm per team
🥉 6th to 24th: 1 LeKiwi per team
🥉 25th to 30th: 1 SO-101 per team

In collaboration with Seed Studio, a total prize value of €15,000, consisting of robotics equipment and related hardware, will be awarded to the winning teams.



1. Imitation Learning

1.1. Real world imitation learning

Why Choose This Track?

Imitation learning is a core component of the wave that has taken over robotics and AI. Models like SmolVLA show how expert demonstrations from the real world, often messy, diverse, and unstructured, can be turned into a robust foundation for training powerful robot policies. Once finetuned, SmolVLA is able to replicate demonstrated behaviors autonomously, allowing robots to perform complex tasks across varied environments.

Choosing this track will help you understand better how this technology works and how it can be used to automate your own tasks.

Programming difficulty	⭐⭐
Need for creativity	⭐⭐⭐
Hardware requirements	⭐

What You'll Do

➡️ *Participants will train their own robot to autonomously perform a new and original real world task with imitation learning*.

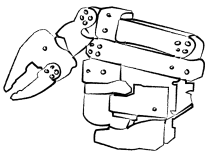
Our robots have already been deployed across a wide range of tasks, but this is just the beginning. New tasks call for creativity, bold experimentation, and a willingness to push the boundaries of what robots can do. This is where you step in: can you imagine a new and original task for our robots to take on?

First, you will set up your robot (SO-100, SO-101, Koch, LeKiwi, etc.) and follow the "Getting Started with a Real-World Robot" tutorial to learn how to record a dataset, train a model and deploy it on your robot. You can start with a simple policy as ACT, and if you have enough resources (time & compute), start experimenting on smolVLA.

You don't have a robot supported by LeRobot but you're still interested in imitation learning ? This track is fully compatible with simulation environments, allowing you to explore and develop without requiring any additional hardware !

Project Ideas

Come up with a creative real world task that one of the robots that are supported can perform. To not limit your creativity we didn't write down concrete examples here but it could be anything the robot physically can perform !



If you feel like you are already confident enough with building your dataset and training a model, here are a few ideas you can dig further :

- 💭 *Asynchronous inference* - Introduced along with smolVLA it allows for a more dynamic and a better adaptation of the robot actions, even with a remote inference server. What about testing it on your own model ?

- 💭 *Online corrections* - With the new SO-101 leader arm, it is now possible to preempt the follower when evaluating a policy. This feature can be used to create new datasets accounting for situations that were not correctly encapsulated by the initial learning dataset;

- 💭 *Data augmentation* - An alternative to create more robust and diverse datasets, whether it be on real world or simulated data.

Hardware Requirements (mandatory)

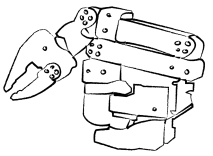
- ☐ A leader robot and a follower robot supported by LeRobot: SO-100, SO-101, Koch, LeKiwi
- ☐ Several cameras: smartphone and laptop cameras, USB webcams, etc.
- ☐ Access to a local GPU or to Google Collab

Resources

- 💡 Getting Started with a Real-World Robot
- 💡 Imitation learning on Real-World robots
- 💡 Imitation learning in Simulation
- 💡 Finetune SmolVLA
- 💡 Asynchronous inference PR
- 💡 Notebook to train ACT on Google Collab **without** Hugging Face GPU credits *Don't forget to clone the notebook before using it !*
- 💡 Notebook to train smolVLA on Google Collab **without** Hugging Face GPU credits *Don't forget to clone the notebook before using it !*
- 💡 Training LeRobot on a Hugging Face compute space **with** Hugging Face GPU credits

Advisors

- 📖 From Hugging Face: Dana Aubakirova (dana_55517), Francesco Capuano (fracapuano), Steven Palma (imstevenpm)
- 📖 From the community: Ville Kuosmanen (v_bananasocks), Alexandre Chapin (alexc1342), Marina Barannikov (mnabnv)



2. Reinforcement Learning

2.1. HIL-SERL in simulation

Why Choose This Track?

You don’t have an SO-100/SO-101 with you for the hackathon, but you’re excited about reinforcement learning? You love diving deep into source code and implementation details? Then this track is for you ☺

Ever wanted to train a virtual robot to perform complex tasks with near-perfect precision in just minutes? With Human-In-the-Loop Reinforcement Learning (HIL-RL) in simulation, you can achieve 100% success rates for manipulation tasks !

This approach dramatically improves sample efficiency. After only a few minutes of human-in-the-loop training, your policy may already be executing the task autonomously ! The track is beginner-friendly yet very satisfying: you’ll watch your AI trainee go from flailing to fluent in real time.

Programming difficulty	⭐⭐
Need for creativity	⭐⭐⭐
Hardware requirements	⭐

What You’ll Do

➡️ *Participants will use the LeRobot HIL-SERL implementation to train robotic policies in simulation.*

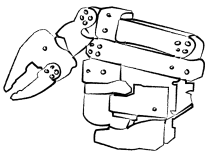
HIL-SERL combines human demonstrations and real-time interventions with reinforcement learning to rapidly learn robust behaviors.

You’ll experiment freely: no hardware required, no risk of damaging a robot, and instant feedback on your ideas. Using the Gym-HIL suite from Hugging Face, you’ll teleoperate a simulated robot (*e.g.* a virtual Franka Panda arm) to provide demonstrations and corrective feedback. You'll start with provided environments and pretrained HIL-SERL models, then innovate with your own ideas.

Project Ideas

1. Use a Pretrained Policy

Start with a pretrained HIL-SERL policy as a baseline (*e.g.* the pick-and-lift task), and adapt it to handle variations, such as different cube starting positions.



2. Extend the Gym-HIL Suite

Add new tasks such as:

- 💭 Pushing a cube into a notch
- 💭 Insertion tasks
- 💭 Pick-and-move tasks

3. Improve the Visual Backbone

The current experiments use a pretrained ResNet-10 image encoder. You could experiment with stronger open-source vision models available on the Hugging Face Hub to speed up training and improve generalization.

4. Simulate SO-101 Tasks

Implement SO-101 tasks in simulation using Gym-HIL and train a policy end-to-end.

5. Design intermediate reward and demonstrate their usefulness

Hardware Requirements

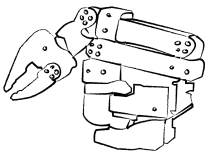
- ☐ Access to a local NVIDIA GPU (mandatory)
- ☐ A system supporting MuJoCo (mandatory)
- ☐ A Linux-based system (recommended)
- ☐ A gamepad for interventions (strongly recommended)
  ⚠️ Keyboard teleoperation is not supported on MacOS ⚠️

Resources

- 💡 HIL-SERL Real Robot Training Workflow Guide
- 💡 Train RL in Simulation
- 💡 HIL-SERL paper

Advisors

- 📖 From Hugging face: Michel Aractingi (michel.aractingi), Adil Zouitine (__boringguy__)
- 📖 From the community: Khalil Meftah (_lilkm), Eugene Mironov (eugenemironov), Ke Wang (kewang1250)



2.2. Real world HIL-SERL (SO-100/SO-101)

Why Choose This Track?

Want to train your first reinforcement learning policy on a real robot, with no simulation or sim2real transition required? This track is for you ☺

Ever wanted to observe your robot progressively learning a new task through the guidance of your corrections ? With real world Human-In-the-Loop Reinforcement Learning (HIL-RL), you'll get the chance to witness an embodied aha moment !

As for simulation, this approach dramatically improves sample efficiency. After a few minutes of human-in-the-loop training, you may start noticing the first signs of autonomous learning and adaptation in your policy. The track involves much more hands-on work, but is highly rewarding: you'll follow your robot trainee progression in real time.

Programming difficulty	⭐⭐⭐
Need for creativity	⭐
Hardware requirements	⭐⭐

What You'll Do

➡️ *Participants will teach their own robot to perform a task through human interaction and reinforcement learning.*

You'll first provide a few successful demonstrations to your policy using either a second robotic arm in leader–follower mode, or a gamepad or a keyboard for manual control. The HIL-SERL system helps you record successful and failed episodes while training a reward classifier to recognize success based on your demonstrations.

Then, you'll launch the HIL-SERL training loop and the robot will begin to perform the task on its own. Your job? Intervene when needed to guide the robot away from mistakes - reinforcement learning takes care of the rest.

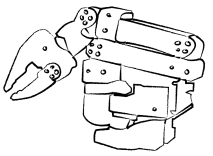
Project Ideas

Start with basic manipulation tasks, such as:

- 👉 Pick-and-lift (a cube or ball from the table)
- 👉 Push to target
- 👉 Press a button

Once you have a working policy, you can try:

1. *Integrating pretrained Vision-Language Actions architectures*



2. Improving the reward model

Move from sparse to shaped or intermediate rewards to better handle long-horizon tasks.

3. Creating general reward models that work across multiple tasks

4. Tackling complex manipulation tasks

Example: block stacking (requires two-stage precision and sub-task structuring).

5. Upgrading the image encoder baseline

Currently uses ResNet-10 - Experiment with open-source vision models from the Hugging Face Hub.

6. Improve automatic take-over with the So101 leader arm.

Current modes of teleoperation in our RL stack contain a `leader_automatic` mode. The goal is to be able to automatically grab the leader arm while it's tracking the follower and intervene. Your task would be to improve this function to further make online learning smoother.

Hardware Requirements

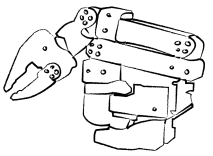
- ☐ Access to a local NVIDIA GPU (mandatory)
- ☐ A Linux-based system (recommended)
- ☐ Several cameras: smartphone and laptop cameras, USB webcams, etc. (mandatory)
- ☐ A leader robot and a follower robot supported by LeRobot: SO-100 or SO-101(follower robot mandatory, leader robot recommended)
- ☐ A gamepad for interventions (strongly recommended with only a follower robot)

  ⚠️ Keyboard teleoperation is not supported on MacOS ⚠️

Resources

- 💡 Getting Started with a Real-World Robot
- 💡 HIL-SERL Real Robot Training Workflow Guide
- 💡 HIL-SERL paper

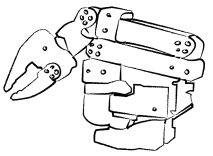
Advisors



📖 From Hugging face: Michel Aractingi (michel.aractingi), Adil Zouitine (__boringguy__)

📖 From the community: Khalil Meftah (_lilkm), Eugene Mironov (eugenemironov), Ke Wang (kewang1250)

3. Create and integrate a new/existing robot/teleoperator to LeRobot



Why Choose This Track?

Adding support for capable robots enables the open source community to extend their use of LeRobot and to experiment cutting-edge algorithms. With this track, we want to challenge the participants to further extend LeRobot scope by integrating a new robot or teleoperator among its supported platforms. This kind of project is especially convinient because it gives you the flexibility to either design and build a new robot from scratch or integrate an existing one into the LeRobot ecosystem.

This track is a great opportunity to dive deeper into the foundations of robotics, exploring key areas like mechanics, electronics, control systems. Furthermore, the integration part will provide you with a deeper understanding of LeRobot codebase and its brand new middleware.

If you feel like robots are not your thing, you might consider focusing on developing a teleoperation system instead !

Programming difficulty	⭐
Need for creativity	⭐⭐
Hardware requirements	⭐⭐⭐

What You'll Do

➡️ *Participants will integrate a newly created or existing robot/teleoperator in LeRobot library.*

If you follow this track, you will either :

👉 Create your own robot and make it compatible with LeRobot

Design your own robot with Feetech or Dynamixel motors (already supported in LeRobot) and show off its capabilities using the LeRobot library.

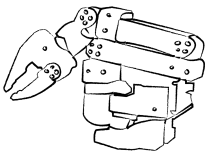
👉 Add an existing robot to LeRobot library

Add an existing robot to LeRobot by creating a new motorbus class (following Feetech and Dynamixel examples) and a new class for the robot, inheriting from the `Robot` base class. If you have time you can even try to carefully teleoperate the robot or even record a dataset !

👉 Explore teleoperating devices (teleoperators)

Create a teleoperation device and/or add an existing teleoperation device to LeRobot by creating a new class for the teleoperator, inheriting from the `Teleoperator` base class.

Project Ideas



There are many widespread robotic platforms waiting to be ported on LeRobot, and even more new robots to design, build and integrate as well ! As this track encompasses a wide diversity of robots, we didn't want to constrain the possibilities to a finite bullet points list.

On the teleoperation devices side however, here are a few accessible ideas to investigate:

- 💭 Use the IMU embedded in every modern smartphones;
- 💭 Use a VR set, a keyboard, a game controller,...
- 💭 Use a digital twin running in simulation.

Hardware Requirements

Creating a new robot/teleoperator ?

- ☐ Access to a 3D CAD software and a 3D printer
- ☐ Motors for the robot (Dynamixel/Feetech)
- ☐ Sensors for the teleoperator

Using an existing robot/teleoperator ?

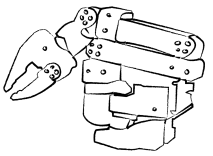
- ☐ The robot/teleoperator itself !

Resources

- 💡 Getting Started with a Real-World Robot
- 💡 Bring your Own Hardware - Adding new hardware in LeRobot
- 💡 Links to the Robot and MotorBus classes implementation
- 💡 SO-arms and LeKiwi CAD files and BOMs *starting point for a new robot*
- 💡 HopeJr CAD files and BOM *starting point for a new robot*
- 💡 *Teleoperation using a VR set* *idea for a new teleoperator*
- 💡 Teleoperation using a digital twin *idea for a new teleoperator*

Advisors

- 📖 From Hugging face: Simon Alibert (syx6820), Pepijn Kooijmans (pepijn8041), Martino Russi (nepyope.)



4. Hardware improvement

Why Choose This Track?

Opening the door for more people to own robots and experiment with automation broadens participation in the robotics ecosystem, and accelerates the improvement of existing robots. Enhanced mechanical and electrical designs, additional sensors and modular features - There are numerous ways to make open-sourced robots more efficient, and more flexible for emerging tasks and novel use cases.

By choosing this track, you're going to join this community effort yourself, and to put forward an hardware improvement on an existing open source robot design (SO-arms, Koch, LeKiwi).

Programming difficulty	⭐
Need for creativity	⭐⭐⭐
Hardware requirements	⭐⭐

What You'll Do

➡️ *Participants will suggest and implement a hardware improvement on one of the robot designs supported in LeRobot library.*

In this track, you'll be challenged to come up with your own original hardware improvement to one of LeRobot supported robot models. This could mean tuning its performance, adding new features, or adapting it to a new use case.

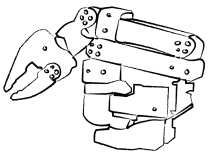
As these models are open-source, you may change *anything*:

- 👉 Tweak the electrical and mechanical designs: new motors, new materials, new 3D models,...
- 👉 Create additional attachments or tools such as camera mounts or grippers.
- 👉 Add new sensors with the appropriate attachment, communication tools and power supply.

Project Ideas

The projects of this track can go as far as your creativity goes. Here's a few ideas if you're struck with a writer's block:

1. *Investigate hardware improvements on LeKiwi*



Because it combines mobility with object manipulation, LeKiwi offers much room for hardware improvements :

- 💭 Create a new open-source omniwheel design;
- 💭 Improve cable routing, communication performances and power management;
- 💭 Improve the teleoperation experience with an overview camera on the back.

2. *Create a versatile camera mount suitable for multiple robots and cameras*

And with other features : different view angles and positions, automatic light adaptation, implicit stereo-vision,...

3. *Add new sensors to LeRobot !*

The community has been asking for it since almost day one - It is time to bring in new sensors in LeRobot library ! Here's a few ideas to find inspiration :

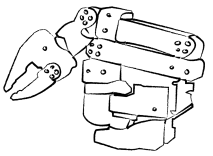
- 💭 Depths cameras, LIDARs and other sensors to get depth frames or point clouds;
- 💭 Microphones (contact microphones, MEMS, electret microphones);
- 💭 Tactile and force sensors;
- 💭 Event cameras.

Hardware Requirements

- ☐ A robot supported by LeRobot: SO-100, SO-101, Koch, LeKiwi (mandatory)
- ☐ Access to a 3D CAD software and a 3D printer (if needed)
- ☐ New sensors (if needed)

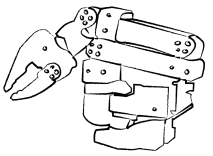
Resources

- 💡 Getting Started with a Real-World Robot
- 💡 Documentation on cameras in LeRobot *for new cameras and sensors integration*
- 💡 An example showing how to add a camera view to a dataset *for new cameras and sensors integration*
- 💡 Adding microphones and audio in LeRobot PR
  ⚠️ *Experimental - Contact Caroline on Discord (nahkriin_caroline)* ⚠️
- 💡 SO-arms and LeKiwi CAD files and BOMs
- 💡 HopeJr CAD files and BOM



Advisors

- 📖 From Hugging Face: Caroline Pascal (nahkriin_caroline), Pepijn Kooijmans (pepijn8041)



5. Datasets tools improvement

Why Choose This Track?

Datasets are the foundation of embodied AI, as they enable models to learn from sensory experiences and interactions within physical or virtual environments. High-quality datasets provide the multimodal information necessary for robots to perceive, reason, and act effectively. Improving and creating tools to streamline data collection, simplify dataset curation and boarden recorded modalities is crucial to train robust, generalizable, and adaptive models.

With this track you're going to delve into the range of possibilities for improving the data—whether in terms of the data itself or the user experience.

Programming difficulty	⭐⭐
Need for creativity	⭐⭐
Hardware requirements	⭐⭐

What You'll Do

➡️ *Participants will create or improve a tool supporting the creation or enhancement of datasets.*

Tools related to datasets are diverse, and may be improved in several ways. For this hackathon, we would like to stress the following leads :

- 👉 Make data collection easier, more efficient, and less alienating;
- 👉 Simplify datasets modification and enhancement thanks to edition or data augmentation tools;
- 👉 Diversify recordable modalities by adding support for new sensors and data types.

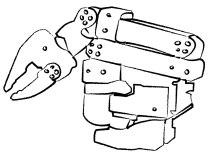
Once you've identified the track that resonates most with you, you'll be asked to propose a new tool or improve an existing one to address a challenge related to dataset creation.

Project Ideas

1. *Make the recording of datasets funnier or more efficient*

For instance, one could train another robot to automatically reset the training environment between two episodes recordings, or guide the user towards unseen situations to generate the next episode initial setting.

2. *Streamline dataset edition in an informed way*



The ultimate goal of this project would be to GUI (e.g. a HF space) to facilitate dataset editing (delete/crop episode, merge datasets, rename observations, etc.) and data augmentation. It could then be improved with a tool to evaluate the quality of an episode inside a dataset.

3. *Improve hardware interfaces*

Tired of CLI tools ? Create a HF space in which you could connect a robot using Webusb and this cool library created by Tim Qian from the community: feetech.js !

4. *Add new sensors support in LeRobotDataset*

Following up on track 4, adding new sensors to LeRobot library also means adding support for the recorded modalities, including (but not limited to):

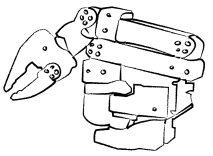
- 💭 Depth frames support;
- 💭 Point clouds support;
- 💭 Time series support (audio data, force/vibration/tactile sensors, etc.);
- 💭 Event-based data support;

Hardware Requirements

- ☐ A leader robot and a follower robot supported by LeRobot: SO-100, SO-101, Koch, LeKiwi (if needed)
- ☐ New sensors (if needed)

Resources

- 💡 LeRobot datasets visualizer
- 💡 Getting Started with a Real-World Robot *(if needed)*
- 💡 Documentation on cameras in LeRobot *for new cameras and sensors integration (robot side)*
- 💡 An example showing how to add a camera view to a dataset *for new cameras and sensors integration (robot side)*
- 💡 Link to the Camera class implementation
- 💡 Link to LeRobotDataset class implementation
- 💡 A GUI for deleting episodes/frames in a dataset
- 💡 Information on how to create a HF space
- 💡 Javascript library you can use in HF spaces to communicate with motors: feetech.js
- 💡 An interesting interpretability visualization tool



Advisors

- 📖 From Hugging Face: Caroline Pascal (nahkriin_caroline), Pepijn Kooijmans (pepijn8041), Steven Palma (imstevenpm)
- 📖 From the community: Ville Kuosmanen (v_bananasocks)