OpenVLA - An Open-Source Vision-Language-Action Model for Robots

Fahd Mirza
50.9K subscribers

Join Subscribe 84 Share Ask Save

5,625 views Jun 14, 2024 #openvla
This video introduces OpenVLA which is a 7B parameter open-source vision-language-action model (VLA), pretrained on 970k robot episodes from the Open X-Embodiment dataset.

@leptok3736 1 year ago
Awesome. Ive been tinkering with something similar since the RT-2 paper came out using Llava. The neat thing i think is making really dumb and simple hardware intelligent. Those simple dc motor cars, cam + pi client. Simple commands like Forward 1.5, rotate_right 0.3 etc. I wonder if i can just prompt it to use those commands instead. Might work better than native vision language models. Have it record successful sequences to generate more data

@pasanmanula6758 1 year ago
Looks like this openvla/openvla-7b is a very memory intensive model. I was trying to run this model on my local machine (which has only 4 gb of GPU memory) and failed. Do you know any smaller model version to test this out?

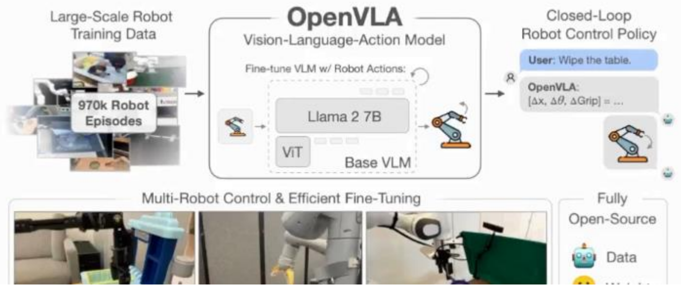
OpenVLA: An Open-Source Vision-Language-Action Model

Moo Jin Kim*[1], Karl Pertsch*[1,2], Siddharth Karamcheti*[1,3], Ted Xiao[4], Ashwin Balakrishna[3], Suraj Nair[3], Rafael Rafailov[1], Ethan Foster[1], Grace Lam, Pannag Sanketi[4], Quan Vuong[5], Thomas Kollar[3], Benjamin Burchfiel[3], Russ Tedrake[3,6], Dorsa Sadigh[1], Sergey Levine[2], Percy Liang[1], Chelsea Finn[1],

*Equal contribution

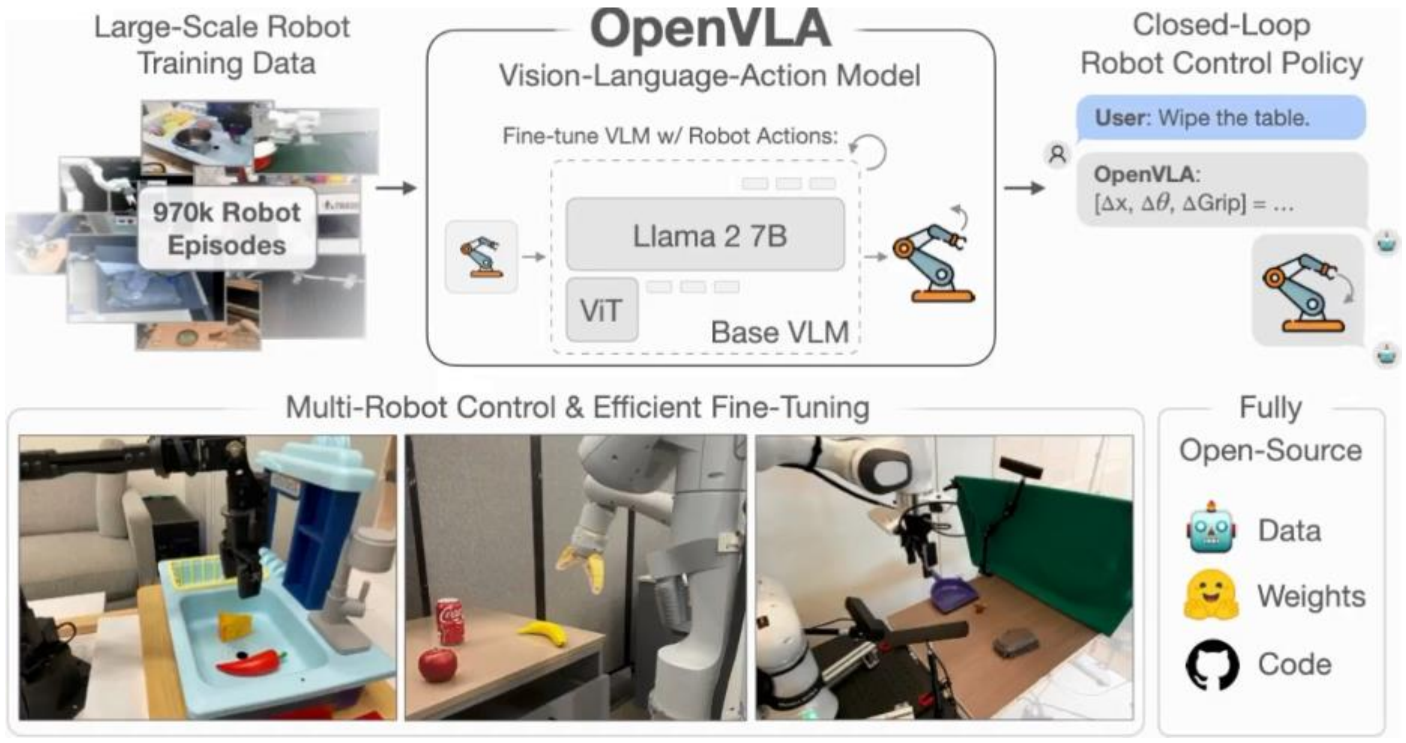
[1]Stanford University, [2]UC Berkeley, [3]Toyota Research Institute, [4]Google DeepMind, [5]Physical Intelligence, [6]MIT,

Paper Code Models

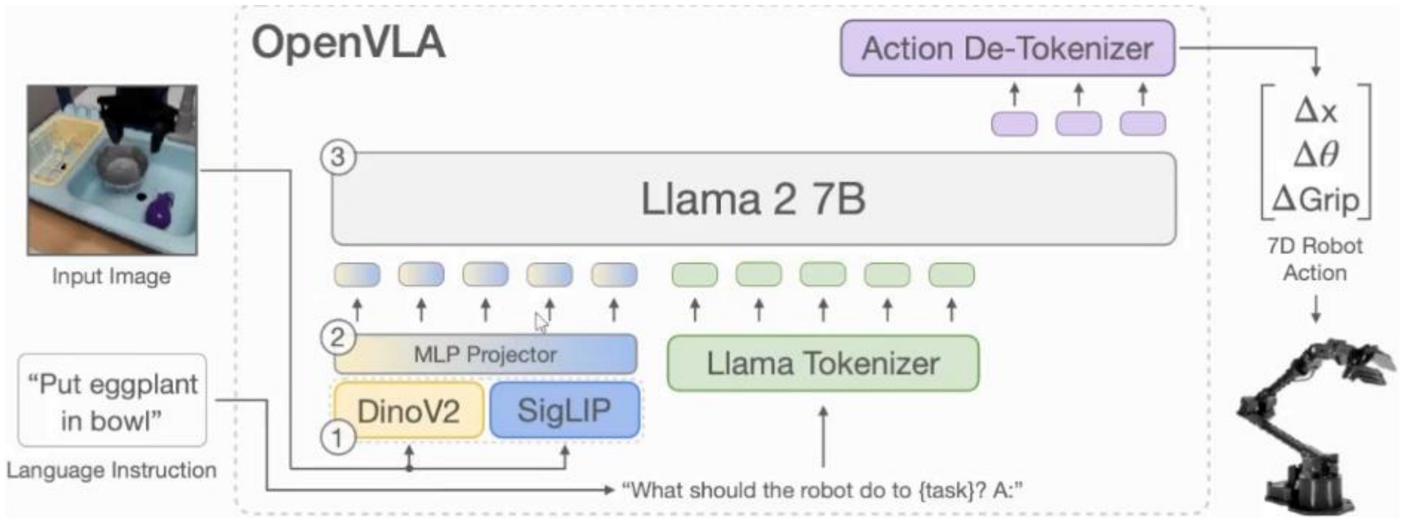

Large-Scale Robot Training Data
970k Robot Episodes
OpenVLA
Vision-Language-Action Model
Fine-tune VLM w/ Robot Actions:
Llama 2 7B
ViT
Base VLM
Closed-Loop Robot Control Policy
User: Wipe the table.
OpenVLA: [Δx, Δθ, ΔGrip] = ...
Multi-Robot Control & Efficient Fine-Tuning
Fully Open-Source
Data

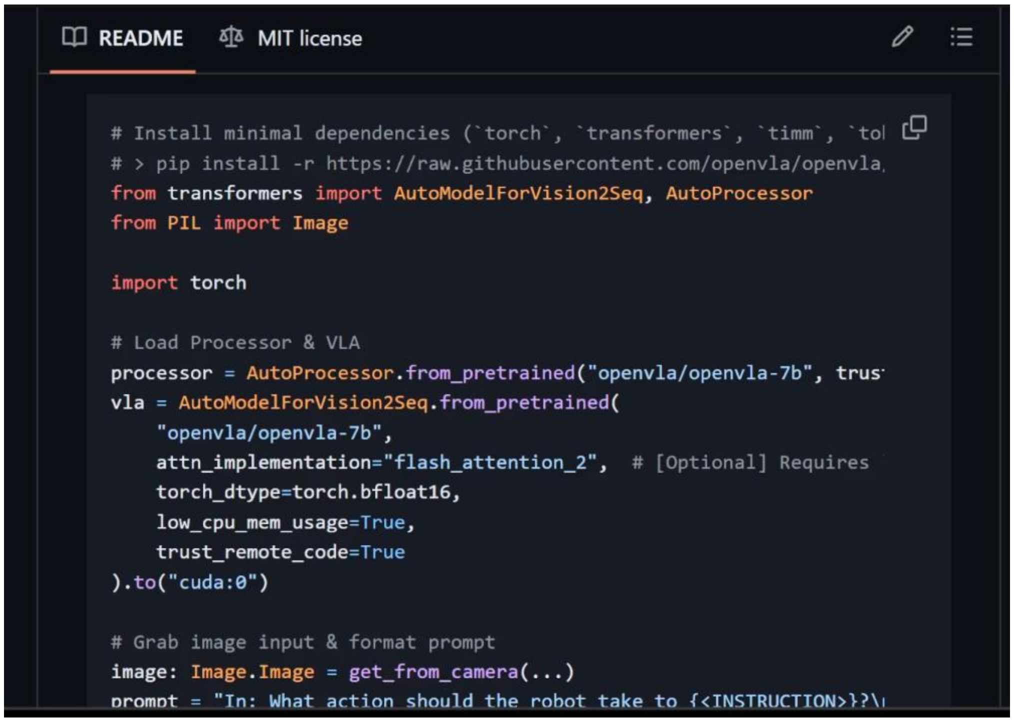
Similarly, we assess the policy's language grounding by prompting it to manipulate different target objects given the same initial states. We find that OpenVLA reliably targets the correct object in most cases.

OpenVLA properly orients the robot's end-effector to align with the orientation of the target object before grasping it.


Large-Scale Robot Training Data
970k Robot Episodes
OpenVLA
Vision-Language-Action Model
Fine-tune VLM w/ Robot Actions:
Llama 2 7B
ViT
Base VLM
Closed-Loop Robot Control Policy
User: Wipe the table.
OpenVLA: [Δx, Δθ, ΔGrip] = …
Multi-Robot Control & Efficient Fine-Tuning
Fully Open-Source
Data
Weights
Code

This OpenVLA model was trained on a cluster of 64 H100s for 15 days, the trained model checkpoints can be downloaded from HF. You can install this model if you have access to a robot arm.





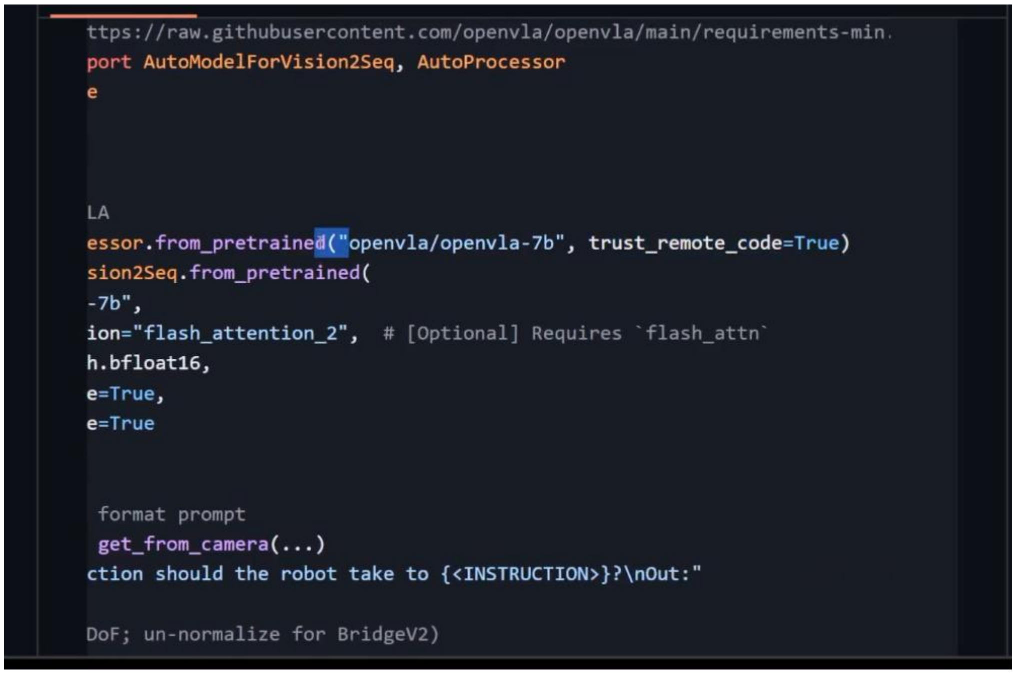
```
# Install minimal dependencies (`torch`, `transformers`, `timm`, `tol
# > pip install -r https://raw.githubusercontent.com/openvla/openvla,
from transformers import AutoModelForVision2Seq, AutoProcessor
from PIL import Image

import torch

# Load Processor & VLA
processor = AutoProcessor.from_pretrained("openvla/openvla-7b", trus
vla = AutoModelForVision2Seq.from_pretrained(
    "openvla/openvla-7b",
    attn_implementation="flash_attention_2",  # [Optional] Requires
    torch_dtype=torch.bfloat16,
    low_cpu_mem_usage=True,
    trust_remote_code=True
).to("cuda:0")

# Grab image input & format prompt
image: Image.Image = get_from_camera(...)
prompt = "In: What action should the robot take to {<INSTRUCTION>}?\
```

Clone and run this in a virtual environment, install flash-attention to speed things up and the deps. You can run this command to control your robot arm.



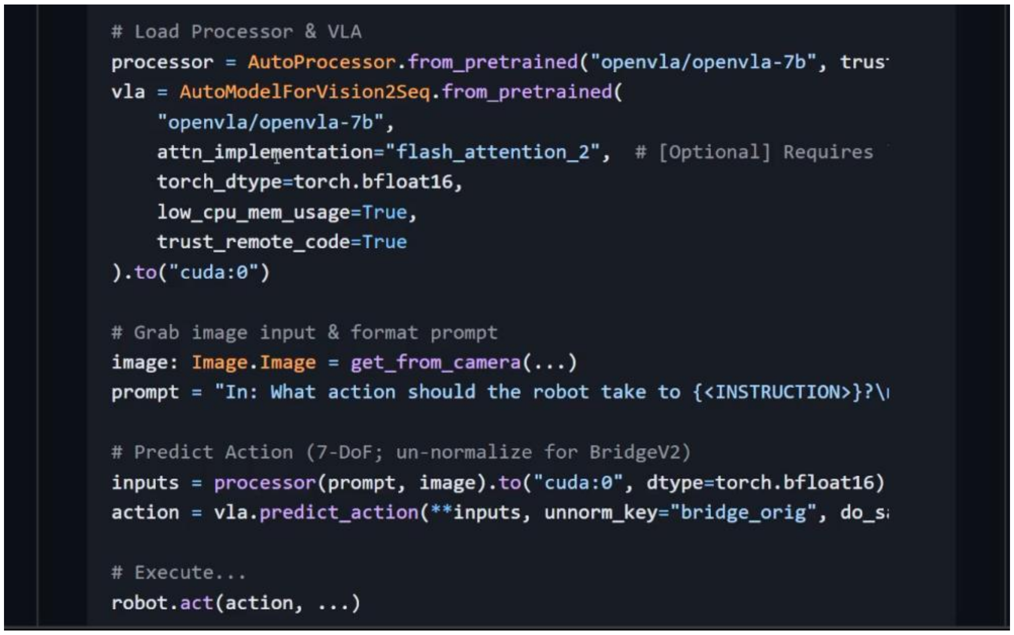
```
ttps://raw.githubusercontent.com/openvla/openvla/main/requirements-min.
port AutoModelForVision2Seq, AutoProcessor
e



LA
essor.from_pretrained("openvla/openvla-7b", trust_remote_code=True)
sion2Seq.from_pretrained(
-7b",
ion="flash_attention_2",  # [Optional] Requires `flash_attn`
h.bfloat16,
e=True,
e=True


 format prompt
 get_from_camera(...)
ction should the robot take to {<INSTRUCTION>}?\nOut:"

DoF; un-normalize for BridgeV2)
```



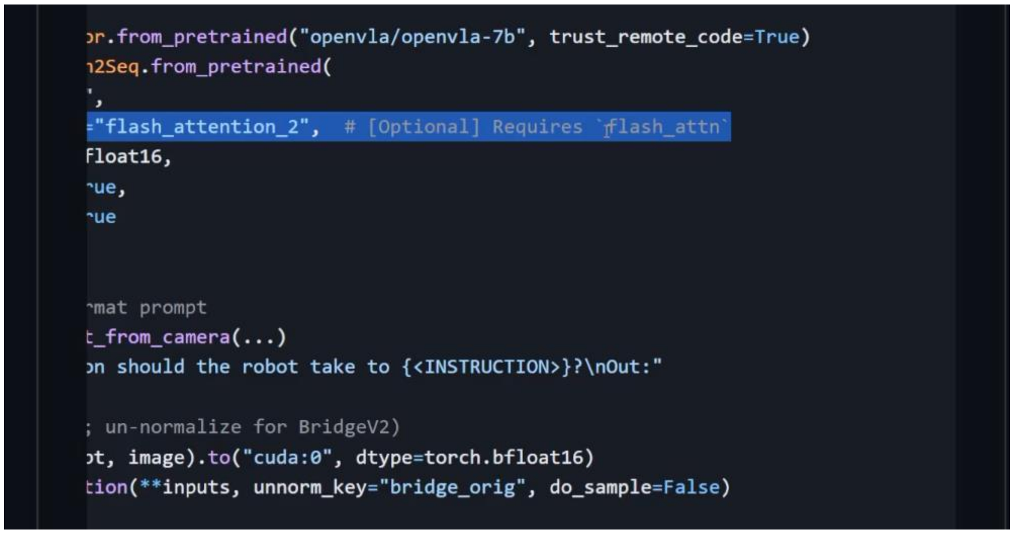
```
# Load Processor & VLA
processor = AutoProcessor.from_pretrained("openvla/openvla-7b", trus
vla = AutoModelForVision2Seq.from_pretrained(
    "openvla/openvla-7b",
    attn_implementation="flash_attention_2",  # [Optional] Requires
    torch_dtype=torch.bfloat16,
    low_cpu_mem_usage=True,
    trust_remote_code=True
).to("cuda:0")

# Grab image input & format prompt
image: Image.Image = get_from_camera(...)
prompt = "In: What action should the robot take to {<INSTRUCTION>}?\

# Predict Action (7-DoF; un-normalize for BridgeV2)
inputs = processor(prompt, image).to("cuda:0", dtype=torch.bfloat16)
action = vla.predict_action(**inputs, unnorm_key="bridge_orig", do_s

# Execute...
robot.act(action, ...)
```

The robot is then going to predict the next action.



```
or.from_pretrained("openvla/openvla-7b", trust_remote_code=True)
n2Seq.from_pretrained(
',
="flash_attention_2",  # [Optional] Requires `flash_attn`
float16,
rue,
rue


rmat prompt
t_from_camera(...)
on should the robot take to {<INSTRUCTION>}?\nOut:"

; un-normalize for BridgeV2)
pt, image).to("cuda:0", dtype=torch.bfloat16)
tion(**inputs, unnorm_key="bridge_orig", do_sample=False)
```