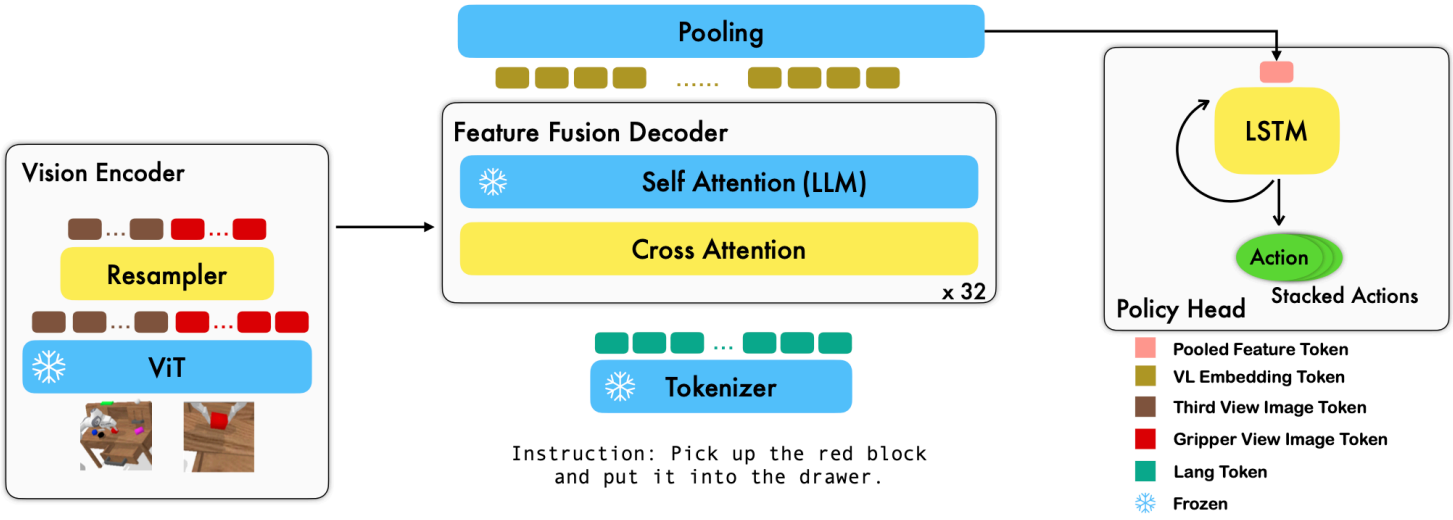
RoboFlamingo

Vision-Language Foundation Models as Effective Robot Imitators

<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" width="76" height="20" ro

arXiv Paper

Hugging Face



RoboFlamingo is a pre-trained-VLM-based robotics learning framework that learns a wide variety of language-conditioned robot skills by fine-tuning on offline free-form imitation datasets.

By exceeding the state-of-the-art performance with a large margin on the CALVIN benchmark, we show that RoboFlamingo can be an effective and competitive alternative to adapt VLMs to robot control.

Our extensive experimental results also reveal several interesting conclusions regarding the behavior of different pre-trained VLMs on manipulation tasks.

RoboFlamingo can be trained or evaluated on a single GPU server (GPU mem requirements depend on the

model size), and we believe RoboFlamingo has the potential to be a cost-effective and easy-to-use solution for robotics manipulation, empowering everyone with the ability to fine-tune their own robotics policy.

This is also the official code repo for the paper Vision-Language Foundation Models as Effective Robot Imitators.

All our experiments are conducted on a single GPU server with 8 Nvidia A100 GPUs (80G).

Pre-trained models are available on Hugging Face.

Usage

Initializing a RoboFlamingo model

We support pre-trained vision encoders from the OpenCLIP package, which includes OpenAI's pre-trained models. We also support pre-trained language models from the `transformers` package, such as MPT, RedPajama, LLaMA, OPT, GPT-Neo, GPT-J, and Pythia models.

```
from robot_flamingo.factor import create_model_and_transforms

model, image_processor, tokenizer = create_model_and_transforms(
    clip_vision_encoder_path="ViT-L-14",
    clip_vision_encoder_pretrained="openai",
    lang_encoder_path="PATH/TO/LLM/DIR",
    tokenizer_path="PATH/TO/LLM/DIR",
    cross_attn_every_n_layers=1,
    decoder_type='lstm',
)
```

The `cross_attn_every_n_layers` argument controls how often cross-attention layers are applied and should be consistent with the VLM. The `decoder_type` argument controls the type of the decoder, currently, we support `lstm`, `fc`, `diffusion` (bugs exist for the dataloader), and `GPT`.

Performance

We report results on the CALVIN benchmark.

Method	Training Data	Test Split	1	2	3	4	5	Avg Len
MCIL	ABCD (Full)	D	0.373	0.027	0.002	0.000	0.000	0.40
HULC	ABCD (Full)	D	0.889	0.733	0.587	0.475	0.383	3.06
HULC (retrained)	ABCD (Lang)	D	0.892	0.701	0.548	0.420	0.335	2.90

Method	Training Data	Test Split	1	2	3	4	5	Avg Len
RT-1 (retrained)	ABCD (Lang)	D	0.844	0.617	0.438	0.323	0.227	2.45
Ours	ABCD (Lang)	D	**0.964**	**0.896**	**0.824**	**0.740**	**0.66**	**4.09**
MCIL	ABC (Full)	D	0.304	0.013	0.002	0.000	0.000	0.31
HULC	ABC (Full)	D	0.418	0.165	0.057	0.019	0.011	0.67
RT-1 (retrained)	ABC (Lang)	D	0.533	0.222	0.094	0.038	0.013	0.90
Ours	ABC (Lang)	D	**0.824**	**0.619**	**0.466**	**0.331**	**0.235**	**2.48**
HULC	ABCD (Full)	D (Enrich)	0.715	0.470	0.308	0.199	0.130	1.82
RT-1 (retrained)	ABCD (Lang)	D (Enrich)	0.494	0.222	0.086	0.036	0.017	0.86
Ours	ABCD (Lang)	D (Enrich)	0.720	0.480	0.299	0.211	0.144	1.85
Ours (freeze-emb)	ABCD (Lang)	D (Enrich)	**0.737**	**0.530**	**0.385**	**0.275**	**0.192**	**2.12**

Prerequisite:

Step 0

Follow the instructions in the OpenFlamingo and CALVIN to download the necessary dataset and VLM pretrained Models.

Download the CALVIN dataset, choose a split with:

```
cd $HULC_ROOT/dataset
sh download_data.sh D | ABC | ABCD | debug
```

Download the released OpenFlamingo models:

# params	Language model	Vision encoder	Xattn interval*	COCO 4-shot CIDEr	VQAv2 4-shot Accuracy	Avg Len	Weights
3B	anas-awadalla/mpt-1b-redpajama-200b	openai CLIP ViT-L/14	1	77.3	45.8	3.94	Link
3B	anas-awadalla/mpt-1b-redpajama-200b-dolly	openai CLIP ViT-L/14	1	82.7	45.7	4.09	Link

# params	Language model	Vision encoder	Xattn interval*	COCO 4-shot CIDEr	VQAv2 4-shot Accuracy	Avg Len	Weights
4B	togethercomputer/RedPajama-INCITE-Base-3B-v1	openai CLIP ViT-L/14	2	81.8	49.0	3.67	Link
4B	togethercomputer/RedPajama-INCITE-Instruct-3B-v1	openai CLIP ViT-L/14	2	85.8	49.0	3.79	Link
9B	anas-awadalla/mpt-7b	openai CLIP ViT-L/14	4	89.0	54.8	3.97	Link

Replace the `${lang_encoder_path}` and `${tokenizer_path}` of the path dictionary (e.g., `mpt_dict`) in `robot_flamingo/models/factory.py` for each pretrained VLM with your own paths.

Step 1

Clone this repo

```
git clone https://github.com/RoboFlamingo/RoboFlamingo.git
```

Install the required packages:

```
cd RoboFlamingo
conda create -n RoboFlamingo python=3.8
source activate RoboFlamingo
pip install -r requirements.txt
```

Training the model (using DDP):

```
torchrun --nnodes=1 --nproc_per_node=8 --master_port=6042 robot_flamingo/train/train_calvin.py \
    --report_to_wandb \
    --llm_name mpt_dolly_3b \
    --traj_cons \
    --use_gripper \
    --fusion_mode post \
    --rgb_pad 10 \
    --gripper_pad 4 \
    --precision fp32 \
    --num_epochs 5 \
    --gradient_accumulation_steps 1 \
    --batch_size_calvin 6 \
    --run_name RobotFlamingoDBG \
    --calvin_dataset ${calvin_dataset_path} \
    --lm_path ${lm_path} \
    --tokenizer_path ${tokenizer_path} \
    --openflamingo_checkpoint ${openflamingo_checkpoint} \
    --cross_attn_every_n_layers 4 \
    --dataset_resampled \
    --loss_multiplier_calvin 1.0 \
    --workers 1 \
    --lr_scheduler constant \
    --warmup_steps 5000 \
    --learning_rate 1e-4 \
    --save_every_iter 10000 \
    --from_scratch \
    --window_size 12 > ${log_file} 2>&1
```

`${calvin_dataset_path}` is the path to the CALVIN dataset;

`${lm_path}` is the path to the pre-trained LLM;

`${tokenizer_path}` is the path to the VLM tokenizer;

`${openflamingo_checkpoint}` is the path to the OpenFlamingo pre-trained model;

`${log_file}` is the path to the log file.

We also provide `robot_flamingo/pt_run_gripper_post_ws_12_traj_aug_mpt_dolly_3b.bash` to launch the training. This bash finetunes the `MPT-3B-IFT` version of the OpenFlamingo model, which contains the **default** hyperparameters to train the model, and corresponds to the best results in the paper.

Evaluating the model on the CALVIN benchmark

```
python eval_ckpts.py
```

By adding the checkpoint name and directory into `eval_ckpts.py`, the script would automatically load the model and evaluate it. For example, if you want to evaluate the checkpoint at path 'your-checkpoint-path', you can modify the `ckpt_dir` and `ckpt_names` variables in eval_ckpts.py, and the evaluation results would be saved as 'logs/your-checkpoint-prefix.log'.

Co-finetune with both robot data (CALVIN) and vision-language data (COCO caption, VQA)

The results shown below indicate that co-training could preserve most ability of the VLM backbone on VL tasks, while losing a bit of performance on robot tasks.

use

```
bash robot_flamingo/pt_run_gripper_post_ws_12_traj_aug_mpt_dolly_3b_co_train.bash
```

to launch co-train RoboFlamingo with CoCO, VQAV2 and CALVIN. You should update CoCO and VQA paths in `get_coco_dataset` and `get_vqa_dataset` in `robot_flamingo/data/data.py`.

Results on the CALVIN benchmark:

Split	SR 1	SR 2	SR 3	SR 4	SR 5	Avg Len
Co-Train	ABC->D	82.9%	63.6%	45.3%	32.1%	23.4%
Fine-tune	ABC->D	82.4%	61.9%	46.6%	33.1%	23.5%
Co-Train	ABCD->D	95.7%	85.8%	73.7%	64.5%	56.1%
Fine-tune	ABCD->D	96.4%	89.6%	82.4%	74.0%	66.2%
Co-Train	ABCD->D (Enrich)	67.8%	45.2%	29.4%	18.9%	11.7%
Fine-tune	ABCD->D (Enrich)	72.0%	48.0%	29.9%	21.1%	14.4%

Results on VL tasks:

				coco					VQA
	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE_L	CIDEr	SPICE	Acc

				coco					VQA
Fine-tune (3B, zero-shot)	0.156	0.051	0.018	0.007	0.038	0.148	0.004	0.006	4.09
Fine-tune (3B, 4-shot)	0.166	0.056	0.020	0.008	0.042	0.158	0.004	0.008	3.87
Co-Train (3B, zero-shot)	0.225	0.158	0.107	0.072	0.124	0.334	0.345	0.085	36.37
Original Flamingo (80B, fine-tuned)	-	-	-	-	-	-	1.381	-	82.0

Acknowledgment

The logo is generated using MidJourney

This work uses code from the following open-source projects and datasets:

CALVIN

Original: https://github.com/mees/calvin
License: MIT

OpenAI CLIP

Original: https://github.com/openai/CLIP
License: MIT

OpenFlamingo

Original: https://github.com/mlfoundations/open_flamingo
License: MIT

Cite our work:

```
@article{li2023vision,
  title     = {Vision-Language Foundation Models as Effective Robot Imitators},
  author    = {Li, Xinghang and Liu, Minghuan and Zhang, Hanbo and Yu, Cunjun and Xu, Jie and Wu, Hongtao an
  journal={arXiv preprint arXiv:2311.01378},
  year={2023}
```