🦩 OpenFlamingo

pypi package 2.0.1

Paper | Blog posts: 1, 2 | Demo

Welcome to our open source implementation of DeepMind's Flamingo!

In this repository, we provide a PyTorch implementation for training and evaluating OpenFlamingo models. If you have any questions, please feel free to open an issue. We also welcome contributions!

Table of Contents

Installation

To install the package in an existing environment, run

```
pip install open-flamingo
```

or to create a conda environment for running OpenFlamingo, run

```
conda env create -f environment.yml
```

To install training or eval dependencies, run one of the first two commands. To install everything, run the third command.

```
pip install open-flamingo[training]
pip install open-flamingo[eval]
pip install open-flamingo[all]
```

There are three `requirements.txt` files:

- `requirements.txt`
- `requirements-training.txt`
- `requirements-eval.txt`

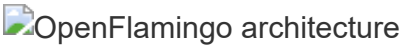
Depending on your use case, you can install any of these with `pip install -r <requirements-file.txt>`. The base file contains only the dependencies needed for running the model.

Approach

OpenFlamingo is a multimodal language model that can be used for a variety of tasks. It is trained on a large multimodal dataset (e.g. Multimodal C4) and can be used to generate text conditioned on interleaved images/text. For example, OpenFlamingo can be used to generate a caption for an image, or to generate a question given an image and a text passage. The benefit of this approach is that we are able to rapidly adapt to new tasks using in-context learning.

Model architecture

OpenFlamingo combines a pretrained vision encoder and a language model using cross attention layers. The model architecture is shown below.



Credit: Flamingo

Usage

Initializing an OpenFlamingo model

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

```
from open_flamingo 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="anas-awadalla/mpt-1b-redpajama-200b",
    tokenizer_path="anas-awadalla/mpt-1b-redpajama-200b",
    cross_attn_every_n_layers=1,
    cache_dir="PATH/TO/CACHE/DIR"  # Defaults to ~/.cache
)
```

Released OpenFlamingo models

We have trained the following OpenFlamingo models so far.

# params	Language model	Vision encoder	Xattn interval*	COCO 4-shot CIDEr	VQAv2 4-shot Accuracy	Weights
3B	mosaicml/mpt-1b-redpajama-200b	openai CLIP ViT-L/14	1	77.3	45.8	Link
3B	mosaicml/mpt-1b-redpajama-200b-dolly	openai CLIP ViT-L/14	1	82.7	45.7	Link
4B	togethercomputer/RedPajama-INCITE-Base-3B-v1	openai CLIP ViT-L/14	2	81.8	49.0	Link

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

** Xattn interval refers to the `--cross_attn_every_n_layers` argument.*

Note: as part of our v2 release, we have deprecated a previous LLaMA-based checkpoint. However, you can continue to use our older checkpoint using the new codebase.

Downloading pretrained weights

To instantiate an OpenFlamingo model with one of our released weights, initialize the model as above and use the following code.

```
# grab model checkpoint from huggingface hub
from huggingface_hub import hf_hub_download
import torch

checkpoint_path = hf_hub_download("openflamingo/OpenFlamingo-3B-vitl-mpt1b", "checkpoint.pt")
model.load_state_dict(torch.load(checkpoint_path), strict=False)
```

Generating text

Below is an example of generating text conditioned on interleaved images/text. In particular, let's try few-shot image captioning.

```
from PIL import Image
import requests
import torch

"""
Step 1: Load images
"""
demo_image_one = Image.open(
    requests.get(
        "http://images.cocodataset.org/val2017/000000039769.jpg", stream=True
    ).raw
)

demo_image_two = Image.open(
    requests.get(
        "http://images.cocodataset.org/test-stuff2017/000000028137.jpg",
        stream=True
    ).raw
)

query_image = Image.open(
    requests.get(
        "http://images.cocodataset.org/test-stuff2017/000000028352.jpg",
        stream=True
    ).raw
)


"""
Step 2: Preprocessing images
Details: For OpenFlamingo, we expect the image to be a torch tensor of shape
 batch_size x num_media x num_frames x channels x height x width.
 In this case batch_size = 1, num_media = 3, num_frames = 1,
 channels = 3, height = 224, width = 224.
"""
vision_x = [image_processor(demo_image_one).unsqueeze(0), image_processor(demo_image_two).unsqueeze(0
vision_x = torch.cat(vision_x, dim=0)
vision_x = vision_x.unsqueeze(1).unsqueeze(0)

"""
Step 3: Preprocessing text
Details: In the text we expect an <image> special token to indicate where an image is.
 We also expect an <|endofchunk|> special token to indicate the end of the text
 portion associated with an image.
"""
```

```
tokenizer.padding_side = "left" # For generation padding tokens should be on the left
lang_x = tokenizer(
    ["<image>An image of two cats.<|endofchunk|><image>An image of a bathroom sink.<|endofchunk|><ima
    return_tensors="pt",
)


"""
Step 4: Generate text
"""
generated_text = model.generate(
    vision_x=vision_x,
    lang_x=lang_x["input_ids"],
    attention_mask=lang_x["attention_mask"],
    max_new_tokens=20,
    num_beams=3,
)

print("Generated text: ", tokenizer.decode(generated_text[0]))
```

Training

We provide training scripts in `open_flamingo/train`. We provide an example Slurm script in `open_flamingo/scripts/run_train.py`, as well as the following example command:

```
torchrun --nnodes=1 --nproc_per_node=4 open_flamingo/train/train.py \
  --lm_path anas-awadalla/mpt-1b-redpajama-200b \
  --tokenizer_path anas-awadalla/mpt-1b-redpajama-200b \
  --cross_attn_every_n_layers 1 \
  --dataset_resampled \
  --batch_size_mmc4 32 \
  --batch_size_laion 64 \
  --train_num_samples_mmc4 125000\
  --train_num_samples_laion 250000 \
  --loss_multiplier_laion 0.2 \
  --workers=4 \
  --run_name OpenFlamingo-3B-vitl-mpt1b \
  --num_epochs 480 \
  --warmup_steps  1875 \
  --mmc4_textsim_threshold 0.24 \
  --laion_shards "/path/to/shards/shard-{0000..0999}.tar" \
  --mmc4_shards "/path/to/shards/shard-{0000..0999}.tar" \
  --report_to_wandb
```

Note: The MPT-1B base and instruct modeling code does not accept the `labels` kwarg or compute cross-entropy loss directly within `forward()`, as expected by our codebase. We suggest using a modified version of the MPT-1B models found here and here.

For more details, see our training README.

Evaluation

An example evaluation script is at `open_flamingo/scripts/run_eval.sh`. Please see our evaluation README for more details.

To run evaluations on OKVQA you will need to run the following command:

```
import nltk
nltk.download('wordnet')
```

Future plans

- [ ] Add support for video input

Team

OpenFlamingo is developed by:

Anas Awadalla*, Irena Gao*, Joshua Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Shiori Sagawa, Jenia Jitsev, Simon Kornblith, Pang Wei Koh, Gabriel Ilharco, Mitchell Wortsman, Ludwig Schmidt.

The team is primarily from the University of Washington, Stanford, AI2, UCSB, and Google.

Acknowledgments

This code is based on Lucidrains' flamingo implementation and David Hansmair's flamingo-mini repo. Thank you for making your code public! We also thank the OpenCLIP team as we use their data loading code and take inspiration from their library design.

We would also like to thank Jean-Baptiste Alayrac and Antoine Miech for their advice, Rohan Taori, Nicholas Schiefer, Deep Ganguli, Thomas Liao, Tatsunori Hashimoto, and Nicholas Carlini for their help with assessing the safety risks of our release, and to Stability AI for providing us with compute resources to train these models.

Citing

If you found this repository useful, please consider citing:

```
@article{awadalla2023openflamingo,
  title={OpenFlamingo: An Open-Source Framework for Training Large Autoregressive Vision-Language Moc
  author={Anas Awadalla and Irena Gao and Josh Gardner and Jack Hessel and Yusuf Hanafy and Wanrong Z
  journal={arXiv preprint arXiv:2308.01390},
  year={2023}
}
```

```
@software{anas_awadalla_2023_7733589,
  author = {Awadalla, Anas and Gao, Irena and Gardner, Joshua and Hessel, Jack and Hanafy, Yusuf and
  title = {OpenFlamingo},
  month        = mar,
  year         = 2023,
  publisher    = {Zenodo},
  version      = {v0.1.1},
  doi          = {10.5281/zenodo.7733589},
  url          = {https://doi.org/10.5281/zenodo.7733589}
}


@article{Alayrac2022FlamingoAV,
  title={Flamingo: a Visual Language Model for Few-Shot Learning},
  author={Jean-Baptiste Alayrac and Jeff Donahue and Pauline Luc and Antoine Miech and Iain Barr and
  journal={ArXiv},
  year={2022},
  volume={abs/2204.14198}
}
```