mPLUG / **mPLUG-Owl3-7B-241101** like 10 Follow mPLUG 97

Visual Question Answering Safetensors English mplugowl3 chat custom_code arxiv:2408.04840 License: apache-2.0

Model card Files xet Community 2

Downloads last month
471

Safetensors Model size 8.07B params Tensor type BF16 Chat template Files info

Inference Providers NEW

Visual Question Answering

This model isn't deployed by any Inference Provider. Ask for provider support

Collection including `mPLUG/mPLUG-Owl3-7B-241101`

Owl Series Collection
7 items • Updated Aug 20

mPLUG-Owl3

Introduction

mPLUG-Owl3 is a state-of-the-art multi-modal large language model designed to tackle the challenges of long image sequence understanding. We propose Hyper Attention, which boosts the speed of long visual sequence understanding in multimodal large language models by sixfold, allowing for processing of visual sequences that are eight times longer. Meanwhile, we maintain excellent performance on single-image, multi-image, and video tasks.

Github: mPLUG-Owl

What's new

`mPLUG-Owl3-7B-241101` is a improved version of `mPLUG-Owl3-7B-240728`.

Fused Hyper Attention

mPLUG-Owl3 requires separate calculations for cross-attention and self-attention, and fuses the outputs of both through a adaptive gate. Now, we use a unified operation that only requires computing attention once.

New template for media inputs

We now use the following format to represent the splited high-resolution images. In addition, we can now enable image splitting when the input consists of multiple images to achieve further performance benefits, which the old version of mPLUG-Owl3 was not trained to handle with this combination.

```
<|start_cut|>2*3
<|image|> <|image|> <|image|>
<|image|> <|image|> <|image|>
<|image|><|end_cut|>
```

And we use the following format to represent video.

```
<|start_video_frame|><|image|><|image|><|image|><|end_video_frame|>
```

Adjusted media_offset

Previously, media_offset recorded the range of images each token could see. During training, since the images from multiple samples are concatenated together along the batch dimension, media_offset needed to be carefully modified, otherwise it would point to the wrong image. To prevent this, media_offset is now a List[List[int]], representing the position of each image in a sample within the batch in the original sequence. This design also makes the computation of the cross-attention mask and MI-Rope more efficient and convenient.

All of these changes are well handled by the processor, and you don't need to change the original way of calling it.

High performance on video and multi-image scenario

Model	NextQA	MVBench	VideoMME w/o sub	LongVideoBench-val	MLVU	LVBench
mPLUG-Owl3-7B-240728	78.6	54.5	53.5	52.1	63.7	-
mPLUG-Owl3-7B-241101	82.3	59.5	59.3	59.7	70.0	43.5

Model	NLVR2	Mantis-Eval	MathVerse-mv	SciVerse-mv	BLINK	Q-Bench2
mPLUG-Owl3-7B-240728	90.8	63.1	65.0	86.2	50.3	74.0
mPLUG-Owl3-7B-241101	92.7	67.3	65.1	82.7	53.8	77.7

Model	VQAv2	OK-VQA	GQA	VizWizQA	TextVQA
mPLUG-Owl3-7B-240728	82.1	60.1	65.0	63.5	69.0
mPLUG-Owl3-7B-241101	83.2	61.4	64.7	62.9	71.4

Model	MMB-EN	MMB-CN	MM-Vet	POPE	AI2D
mPLUG-Owl3-7B-240728	77.6	74.3	40.1	88.2	73.8
mPLUG-Owl3-7B-241101	80.4	79.1	39.8	88.1	77.8

Quickstart

Load the mPLUG-Owl3. We now only support attn_implementation in `['sdpa', 'flash_attention_2']`.

```
import torch
from modelscope import AutoConfig, AutoModel
model_path = 'iic/mPLUG-Owl3-2B-241101'
config = AutoConfig.from_pretrained(model_path, trust_remote_code=True)
print(config)
model = AutoModel.from_pretrained(model_path, attn_implementation='flash_attentior
_ = model.eval().cuda()
device = "cuda"
```

Chat with images.

```
from PIL import Image

from modelscope import AutoTokenizer
from decord import VideoReader, cpu
tokenizer = AutoTokenizer.from_pretrained(model_path)
processor = model.init_processor(tokenizer)

image = Image.new('RGB', (500, 500), color='red')

messages = [
    {"role": "user", "content": """<|image|>
Describe this image."""},
    {"role": "assistant", "content": ""}
]

inputs = processor(messages, images=[image], videos=None)

inputs.to('cuda')
inputs.update({
    'tokenizer': tokenizer,
    'max_new_tokens':100,
    'decode_text':True,
})


g = model.generate(**inputs)
print(g)
```

Chat with a video.

```
from PIL import Image

from modelscope import AutoTokenizer
from decord import VideoReader, cpu    # pip install decord
tokenizer = AutoTokenizer.from_pretrained(model_path)
processor = model.init_processor(tokenizer)



messages = [
    {"role": "user", "content": """<|video|>
Describe this video."""},
    {"role": "assistant", "content": ""}
]

videos = ['/nas-mmu-data/examples/car_room.mp4']

MAX_NUM_FRAMES=16

def encode_video(video_path):
    def uniform_sample(l, n):
        gap = len(l) / n
        idxs = [int(i * gap + gap / 2) for i in range(n)]
        return [l[i] for i in idxs]

    vr = VideoReader(video_path, ctx=cpu(0))
    sample_fps = round(vr.get_avg_fps() / 1)  # FPS
    frame_idx = [i for i in range(0, len(vr), sample_fps)]
    if len(frame_idx) > MAX_NUM_FRAMES:
        frame_idx = uniform_sample(frame_idx, MAX_NUM_FRAMES)
    frames = vr.get_batch(frame_idx).asnumpy()
    frames = [Image.fromarray(v.astype('uint8')) for v in frames]
    print('num frames:', len(frames))
    return frames
video_frames = [encode_video(_) for _ in videos]
inputs = processor(messages, images=None, videos=video_frames)

inputs.to(device)
inputs.update({
    'tokenizer': tokenizer,
    'max_new_tokens':100,
    'decode_text':True,
})


g = model.generate(**inputs)
print(g)
```

Save memory by Liger-Kernel

mPLUG-Owl3 is based on Qwen2, which can be optimized through the Liger-Kernel to reduce memory usage.

```
pip install liger-kernel
```

```
def apply_liger_kernel_to_mplug_owl3(
    rms_norm: bool = True,
    swiglu: bool = True,
    model = None,
) -> None:
    from liger_kernel.transformers.monkey_patch import _patch_rms_norm_module
    from liger_kernel.transformers.monkey_patch import _bind_method_to_module
    from liger_kernel.transformers.swiglu import LigerSwiGLUMLP
    """
    Apply Liger kernels to replace original implementation in HuggingFace Qwen2 mo

    Args:
        rms_norm (bool): Whether to apply Liger's RMSNorm. Default is True.
        swiglu (bool): Whether to apply Liger's SwiGLU MLP. Default is True.
        model (PreTrainedModel): The model instance to apply Liger kernels to, if
        loaded. Default is None.
    """

    base_model = model.language_model.model

    if rms_norm:
        _patch_rms_norm_module(base_model.norm)

    for decoder_layer in base_model.layers:
        if swiglu:
            _bind_method_to_module(
                decoder_layer.mlp, "forward", LigerSwiGLUMLP.forward
            )
        if rms_norm:
            _patch_rms_norm_module(decoder_layer.input_layernorm)
            _patch_rms_norm_module(decoder_layer.post_attention_layernorm)
    print("Applied Liger kernels to Qwen2 in mPLUG-Owl3")

import torch
from modelscope import AutoConfig, AutoModel
model_path = 'iic/mPLUG-Owl3-2B-241101'
config = AutoConfig.from_pretrained(model_path, trust_remote_code=True)
print(config)
model = AutoModel.from_pretrained(model_path, attn_implementation='flash_attentior
_ = model.eval().cuda()
device = "cuda"
apply_liger_kernel_to_mplug_owl3(model=model)
```

Save memory by setting device_map

When you have more than one GPUs, you can set the `device_map='auto'` to split the mPLUG-Owl3 into multiple GPUs. However, it will slowdown the inference speed.

```
model = AutoModel.from_pretrained(model_path, attn_implementation='flash_attentior
_ = model.eval()
first_layer_name = list(model.hf_device_map.keys())[0]
```

```
device = model.hf_device_map[first_layer_name]
```

Citation

If you find our work helpful, feel free to give us a cite.

```
@misc{ye2024mplugowl3longimagesequenceunderstanding,
      title={mPLUG-Owl3: Towards Long Image-Sequence Understanding in Multi-Modal
      author={Jiabo Ye and Haiyang Xu and Haowei Liu and Anwen Hu and Ming Yan anc
      year={2024},
      eprint={2408.04840},
      archivePrefix={arXiv},
      primaryClass={cs.CV},
      url={https://arxiv.org/abs/2408.04840},
}
```