利用 IQA 方法过滤低质量图片

本文将介绍利用 PaddleMIX 中 `paddlemix.datacopilot.ops.filter` 的 IQA 方法过滤数据集里 `低质量` 图片的方法。

IQA 即 Image Quality Assessment，图像质量评估方法，是通过对图像进行特性分析研究，然后评估出图像优劣，如图像失真、扭曲、模糊，以及图像的美学等方面的定量分析方法。

本文将介绍以下几个部分：

- PaddleMIX 的 IQA 方法
- PaddleMIX 的 Filter
- PaddleMIX 的 Tagger

1. PaddleMIX 的 IQA 方法

本文将引入两个 IQA 方法：

- ARNIQA: ARNIQA: Learning Distortion Manifold for Image Quality Assessment
- BRISQUE: No-Reference Image Quality Assessment in the Spatial Domain

1.1 ARNIQA

ARNIQA 在文章 ARNIQA: Learning Distortion Manifold for Image Quality Assessment 中引入。

> ARNIQA 是一种用于图像质量评估（IQA）的自监督学习方法，全称为 "leArning distoRtion maNifold for Image Quality Assessment" 。它的核心目标是开发一种无需高质量参考图像就能测量图像质量的方法，与人类感知一致。ARNIQA 通过建模图像失真流形（distortion manifold）来获得质量表示，这是一种内在的方式。

PaddleMIX 中的 ARNIQA 模型在 `PaddleMIX/paddlemix/datacopilot/nn/arniqa` 目录中。

原模型使用 PyTorch 实现，这里通过 X2Paddle 将原模型转换为 Paddle 模型后使用。

ARNIQA 的结构大体可以分为：

- encoder
- regressor

两部分。

X2Paddle 的转换代码如下：

```
import pickle
from collections import OrderedDict
```

```
import numpy as np

import torch
import torchvision
from torch import nn
from transformers import GPT2Model, GPT2Tokenizer

import paddle
from iqa_arniqa import forward as paddle_forward
from iqa_arniqa_torch import forward as torch_forward

# Part 1. 构建原模型
# 构建 encoder
encoder = torchvision.models.resnet50(
    weights=torchvision.models.ResNet50_Weights.IMAGENET1K_V1)
feat_dim = encoder.fc.in_features

print('-' * 20)
print(encoder)

encoder = nn.Sequential(*list(encoder.children())[:-1])

print('=' * 20)
print(encoder)

print(feat_dim)

# 加载参数：https://hf-mirror.com/chaofengc/IQA-PyTorch-
Weights/blob/main/ARNIQA.pth
encoder_state_dict = torch.load('../dataset/ARNIQA/ARNIQA.pth',
                                weights_only=True)

# 参考 pyiqa 进行参数映射
cleaned_encoder_state_dict = OrderedDict()
for key, value in encoder_state_dict.items():
    # Remove the prefix
    if key.startswith("model."):
        new_key = key[6:]
        cleaned_encoder_state_dict[new_key] = value

encoder.load_state_dict(cleaned_encoder_state_dict)
encoder.eval()

# 构建 regressor：https://hf-mirror.com/chaofengc/IQA-PyTorch-
Weights/blob/main/regressor_koniq10k.pth
regressor: nn.Module = torch.jit.load(
    '../dataset/ARNIQA/regressor_koniq10k.pth'
)  # Load regressor from torch.hub as JIT model
regressor.eval()

# regressor 是个线性模型
print('-' * 20)
print(regressor.biases)
print(regressor.weights)

# 模拟输入
input_data = np.random.rand(1, 3, 256, 256).astype('float32')
```

```
input_model = torch.tensor(input_data)
input_model_paddle = paddle.to_tensor(input_data)
input_model_resnet50 = paddle.to_tensor(input_data)

save_dir = "pd_model"
jit_type = "trace"

from x2paddle.convert import pytorch2paddle

# 转换 encoder
pytorch2paddle(encoder,
               save_dir,
               jit_type, [input_model],
               disable_feedback=True)

input_data_regressor = np.random.rand(1, 2048).astype('float32')
input_regressor = torch.tensor(input_data_regressor)
input_regressor_paddle = paddle.to_tensor(input_data_regressor)

save_dir = "pd_model_regressor"

# 转换 regressor
pytorch2paddle(regressor,
               save_dir,
               jit_type, [input_regressor],
               disable_feedback=True)

IMAGENET_DEFAULT_MEAN = (0.485, 0.456, 0.406)
IMAGENET_DEFAULT_STD = (0.229, 0.224, 0.225)

default_mean = torch.Tensor(IMAGENET_DEFAULT_MEAN).view(1, 3, 1,
1)
default_std = torch.Tensor(IMAGENET_DEFAULT_STD).view(1, 3, 1, 1)

default_mean_paddle =
paddle.to_tensor(IMAGENET_DEFAULT_MEAN).view([1, 3, 1, 1])
default_std_paddle =
paddle.to_tensor(IMAGENET_DEFAULT_STD).view([1, 3, 1, 1])

torch_score = torch_forward(input_model, encoder, regressor,
default_mean,
                            default_std, feat_dim)

# 查看原模型分数
print('-' * 20)
print('>>> torch score:', torch_score)

# Part 2. 加载转换后的模型
from pd_model.x2paddle_code import Sequential as
encoder_paddle_model
from pd_model_regressor.x2paddle_code import TorchLinearRegression
as regressor_paddle_model

paddle.disable_static()

encoder_paddle = encoder_paddle_model()
```

```
regressor_paddle = regressor_paddle_model()

encoder_paddle_params = paddle.load(r'./pd_model/model.pdparams')
encoder_paddle.set_dict(encoder_paddle_params,
use_structured_name=True)
encoder_paddle.eval()

regressor_paddle_params =
paddle.load(r'./pd_model_regressor/model.pdparams')
regressor_paddle.set_dict(regressor_paddle_params,
use_structured_name=True)
regressor_paddle.eval()

paddle_score = paddle_forward(input_model_paddle, encoder_paddle,
                              regressor_paddle,
default_mean_paddle,
                              default_std_paddle, feat_dim)

# 查看转换后计算的分数
print('-' * 20)
print('>>> paddle score:', paddle_score)

# 将 encoder 和 regressor 整合到一个 ARNIQA 模型中
from iqa_arniqa_model import ARNIQA

arniqa = ARNIQA(default_mean_paddle, default_std_paddle, feat_dim)
arniqa_score = arniqa(input_model_paddle)

# 查看整合后计算的分数
print('-' * 20)
print('>>> arniqa score:', arniqa_score)

# 保存整合后的模型
from os import path as osp

input_spec = paddle.static.InputSpec(shape=[-1, 3, -1, -1],
                                     name='x',
                                     dtype='float32')
static_model = paddle.jit.to_static(arniqa,
                                    input_spec=[input_spec],
                                    full_graph=True)
paddle.jit.save(static_model,
                osp.join('pd_model_arniqa',
"inference_model/model"))

paddle.enable_static()
exe = paddle.static.Executor()
[prog, inputs, outputs] = paddle.static.load_inference_model(
    path_prefix="pd_model_arniqa/inference_model/model",
executor=exe)
result = exe.run(prog, feed={inputs[0]: input_data},
fetch_list=outputs)
print('-' * 20)
print('>>> arniqa static score:', result)

# Part 3. 比对 pyiqa 中的 arniqa 模型所计算的分数
import pyiqa
```

```
iqa_metric = pyiqa.create_metric('arniqa')
score_nr = iqa_metric(input_model)

print('-' * 20)
print('>>> pyiqa score:', score_nr)
```

查看打印结果：

```
--------------------
>>> torch score: tensor([0.4777], grad_fn=<AddBackward0>)
--------------------
>>> paddle score: Tensor(shape=[1], dtype=float32,
place=Place(gpu:0), stop_gradient=False,
       [0.47766486])
--------------------
>>> arniqa score: Tensor(shape=[1], dtype=float32,
place=Place(gpu:0), stop_gradient=False,
       [0.47766486])
I1222 06:34:13.690047   227 pir_interpreter.cc:1445] New Executor
is Running ...
I1222 06:34:13.701393   227 pir_interpreter.cc:1471] pir
interpreter is running by multi-thread mode ...
--------------------
>>> arniqa static score: [array([0.47766486], dtype=float32)]
--------------------
>>> pyiqa score: tensor([0.4777], device='cuda:0')
```

可以看到，转换后的模型精度可以对齐原模型。

这里将转换后的 encoder 与 regressor 分别放入 `PaddleMIX/paddlemix/datacopilot/nn/arniqa` 目录中。

1.2 在 PaddleMIX 中使用 ARNIQA

在 PaddleMIX 中可以直接调用 ARNIQA 模型，如：

```
import numpy as np
import paddle
from paddlemix.datacopilot.nn import ARNIQA

arniqa = ARNIQA()
input_data = np.random.rand(1, 3, 256, 256).astype('float32')
score = arniqa(paddle.to_tensor(input_data))
print(score)
```

结果为

```
Tensor(shape=[1], dtype=float32, place=Place(gpu:0),
stop_gradient=False,
       [0.41729081])
```

1.3 BRISQUE

BRISQUE 在文章 No-Reference Image Quality Assessment in the Spatial Domain 中引入。

> BRISQUE（Blind/Referenceless Image Spatial Quality Evaluator）是一种无参考图像质量评估（NR-IQA）算法，由Mittal等人在2012年提出 。这种算法的核心思想是利用自然场景统计（NSS）特征来评估图像质量，而无需任何参考图像。BRISQUE 使用支持向量机（SVM）对提取的特征进行分类，从而预测图像的质量分数。

PaddleMIX 中的 BRISQUE 模型直接使用 brisque 中所提供的接口。

示例如下：

```
from brisque import BRISQUE

obj = BRISQUE(url=False)
score_brisque = obj.score(np.squeeze(input_data).transpose([1, 2,
0]))
print('-' * 20)
print('>>> brisque score:', score_brisque)
```

可以得到结果：

```
--------------------
>>> brisque score: 89.80066752516188
```

这里需要注意的是，brisque 计算的结果越高，则图像质量越差，计算结果通常需要在 `[0, 100]` 之间。

这里为了与 ARNIQA 方法保持一致，需要对其结果进行 `[0, 100]` 之间的裁减并转换至 `[0, 1]` 之间，并且分数越高，则图像质量越好。

转换代码主要逻辑为：

```
def _score(model, img_path):
    img = Image.open(img_path)
    img = np.asarray(img)
    score = model.score(img=img)

    # clip score to [0, 100]
    score = max(0, score)
    score = min(100, score)

    # convert BRISQUE score [0, 100] to [0, 1], and higher is
better
    score = (100 - score)/100
    return score
```

2. PaddleMIX 的 Filter

使用 Filter 可以过滤数据集中特定的 `低质量` 数据，这里的 `低质量` 如果针对图像数据，可以是模糊、扭曲的图像，如果针对文本，可以是长度过短、意义不明的文本，如果针对图像与文本对，可以是图像与文本匹配程度低等情况。

本文引入三个 Filter：

- iqa_arniqa，利用 ARNIQA 模型过滤低质量的图像。
- iqa_brisque，利用 BRISQUE 模型过滤低质量的图像。

- ensemble，根据数据集中已有的 `tagger`，如 ARNIQA 分数与 BRISQUE 分数，与各 `tagger` 的权重组合过滤数据。

其中 `iqa_arniqa` 与 `iqa_brisque` 针对图像数据进行过滤，`ensemble` 可以组合过滤数据。

2.1 过滤器 iqa_arniqa

过滤器 `iqa_arniqa` 在文件 `PaddleMIX/paddlemix/datacopilot/ops/filter/_iqa_arniqa.py` 中。

通过上文提到的 ARNIQA 模型对图像进行打分过滤，接口如下：

```
def iqa_arniqa(
    item: T,
    min_score: float = 0.,
    max_score: float = 1.,
    key: str = 'image',
    ) -> bool:
```

其中：

- item，为输入项
- min_score，为过滤所需的最小分数
- max_score，为过滤所需的最大分数
- key，为 item 中图像的键

返回为 `bool` 值，如果计算得到的分数在 `min_score` 与 `max_score` 之间，则返回 `True`，否则返回 `False`。

另外，如果 `item` 中没有 `key` (如 `image`) 字段，则默认返回 `True`，即保留此数据。

这里抽取 llava_v1_5_mix665k 数据中的 10 张图片用于演示接口的使用。

In [1]:
```
from functools import partial
import numpy as np
import paddle
from paddlemix.datacopilot.core import MMDataset
from paddlemix.datacopilot.ops.filter import iqa_arniqa

path = 'llava_tmp_10.json'
dataset = MMDataset.from_json(path)

fn = partial(iqa_arniqa, min_score=0.6)
new_dataset = dataset.filter(fn, max_workers=1)
```

```
/opt/conda/envs/python35-paddle120-env/lib/python3.10/site-packages/paddle/utils/cpp_extension/extension_utils.py:686: UserWarning: No ccache found. Please be aware that recompiling all source files may be required. You can download and install ccache from: https://github.com/ccache/ccache/blob/master/doc/INSTALL.md
  warnings.warn(warning_message)
/opt/conda/envs/python35-paddle120-env/lib/python3.10/site-packages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
/opt/conda/envs/python35-paddle120-env/lib/python3.10/site-packages/_distutils_hack/__init__.py:26: UserWarning: Setuptools is replacing distutils.
  warnings.warn("Setuptools is replacing distutils.")
[2024-12-22 16:06:43,819] [ WARNING] - Detected that datasets module was imported before paddlenlp. This may cause PaddleNLP datasets to be unavalible in intranet. Please import paddlenlp before datasets module to avoid download issues
W1222 16:06:45.513878 366600 gpu_resources.cc:119] Please NOTE: device: 0, GPU Compute Capability: 7.0, Driver API Version: 12.0, Runtime API Version: 11.8
W1222 16:06:45.515270 366600 gpu_resources.cc:164] device: 0, cuDNN Version: 8.9.
import module error: fast_ln
import module error: fused_ln
Warning, FusedLn module is not available, use LayerNorm instead.
Warning, FusedLn module is not available, use LayerNorm instead.
modeling_internlm2 has_flash_attn is True.
modeling_intern_vit has_flash_attn is True.
paddlenlp is not installed.
100%|██████████| 10/10 [00:02<00:00,  4.13it/s]
```

In [2]:
```python
len(dataset), len(new_dataset)
```

Out[2]:
```
(10, 9)
```

In [3]:
```python
raw_imgs = set()
new_imgs = set()
for i in dataset:
    raw_imgs.add(i['image'])
for i in new_dataset:
    new_imgs.add(i['image'])

print(raw_imgs - new_imgs)
```

```
{'train2017/000000494884.jpg'}
```

In [4]:
```python
from PIL import Image

img = Image.open('train2017/000000494884.jpg')
img.show()
```

可以看到，通过 `iqa_arniqa` 并设置最小分数 `min_score=0.6` ，过滤后，数据集中过滤掉了上面的这张图片，数据集由 `10` 个过滤为 `9` 个。

2.2 过滤器 iqa_brisque

过滤器 `iqa_brisque` 在文件 `PaddleMIX/paddlemix/datacopilot/ops/filter/_iqa_brisque.py` 中。

通过上文提到的 BRISQUE 模型对图像进行打分过滤，接口如下：

```
def iqa_brisque(
    item: T,
    min_score: float = 0.,
    max_score: float = 1.,
    key: str = 'image',
    ) -> bool:
```

此接口的参数与 `iqa_arniqa` 相同，这里同样使用上面从 llava_v1_5_mix665k 抽取到的 10 张图片用于演示接口的使用。

In [5]:
```
from functools import partial
import numpy as np
import paddle
from paddlemix.datacopilot.core import MMDataset
from paddlemix.datacopilot.ops.filter import iqa_brisque

path = 'llava_tmp_10.json'
dataset = MMDataset.from_json(path)

fn = partial(iqa_brisque, min_score=0.7)
new_dataset = dataset.filter(fn, max_workers=1)
```

```
100%|██████████| 10/10 [00:02<00:00,  3.67it/s]
```

In [6]:
```
len(dataset), len(new_dataset)
```

Out[6]:
```
(10, 9)
```

In [7]:
```
raw_imgs = set()
new_imgs = set()
for i in dataset:
    raw_imgs.add(i['image'])
for i in new_dataset:
    new_imgs.add(i['image'])

print(raw_imgs - new_imgs)
```

```
{'train2017/000000504730.jpg'}
```

In [8]:
```
from PIL import Image

img = Image.open('train2017/000000504730.jpg')
img.show()
```

可以看到，通过 `iqa_brisque` 并设置最小分数 `min_score=0.7` ，过滤后，数据集中过滤掉了上面的这张图片，数据集由 `10` 个过滤为 `9` 个。

通过以上演示可以看到， `iqa_arniqa` 与 `iqa_brisque` 方法接口的使用是一样的，但是结果不一样，这对分析数据集的质量提供了多种角度的考虑方式。

另外， `ensemble` 过滤器可已结合后文提到的 `Tagger` 一同使用，后文会详细介绍 `ensemble` 过滤器的使用。

3. PaddleMIX 的 Tagger

通过 Filter 可以过滤掉数据集中的特定数据，但是，当数据集很大的时候，我们需要一种手段先对数据集进行分析评估，然后再对数据集进行过滤采样。

由此，这里引入 `Tagger` ，即数据标注，可以先通过 `Tagger` 对数据集中的数据进行打分，将分数嵌入到数据集中，然后再通过宏观分析，如分数的均值、方差等，决定过滤掉哪些数据，如分数低于均值的数据。

`Tagger` 的接口类位于 `PaddleMIX/paddlemix/datacopilot/ops/filter/_tagger.py` 中：

```
class Tagger(object):

    def tag(self, item: T) -> T:
        item[self.key()] = self.score(item)
        return item

    def key(self) -> str:
        raise NotImplementedError

    def score(self, item: T) -> float:
        raise NotImplementedError

    def __call__(self, item: T) -> T:
        return self.tag(item)
```

主要定义了：

- `tag` 方法，用于返回打完分数的 item
- `key` 方法，需要子类实现，用于区分不同的 Tagger
- `score` 方法，用于对 item 进行打分
- `__call__` 方法，用于直接调用

> 说明：由于 Tagger 与 Filter 作用相近却不同，因此，暂未将两者整合到一起。

本文引入两个 Tagger：

- ARNIQATagger，利用 ARNIQA 模型对数据进行打分
- BRISQUETagger，利用 BRISQUE 模型对数据进行打分

3.1 ARNIQATagger

`ARNIQATagger` 在文件
`PaddleMIX/paddlemix/datacopilot/ops/filter/_iqa_arniqa.py` 中。

可以直接使用 `tag_arniqa` 对数据集打标。

In [9]:
```
from paddlemix.datacopilot.ops.filter import tag_arniqa
tag_dataset = dataset.map(tag_arniqa)
```

```
100%|██████████| 10/10 [00:01<00:00,  8.90it/s]
```

In [10]:
```
# 查看嵌入的字段
print(tag_dataset[0].keys())
```

```
dict_keys(['id', 'image', 'conversations', '__tag_arniqa'])
```

In [11]:
```
# 查看嵌入的分数
for d in tag_dataset:
    print(d['__tag_arniqa'])
```

```
0.64162034
0.68887085
0.7187992
0.674319
0.6732159
0.74029523
0.59954
0.69299656
0.68343085
0.7909594
```

3.2 BRISQUETagger

`BRISQUETagger` 在文件
`PaddleMIX/paddlemix/datacopilot/ops/filter/_iqa_brisque.py` 中。

可以直接使用 `tag_brisque` 对数据集打标。

In [12]:
```
from paddlemix.datacopilot.ops.filter import tag_brisque
tag_dataset = tag_dataset.map(tag_brisque)
```

```
100%|██████████| 10/10 [00:02<00:00,  4.42it/s]
```

In [13]:
```
# 查看嵌入的字段
print(tag_dataset[0].keys())
```

```
dict_keys(['id', 'image', 'conversations', '__tag_arniqa', '__tag_brisqu
e'])
```

In [14]:
```
# 查看嵌入的分数
for d in tag_dataset:
    print(d['__tag_arniqa'], d['__tag_brisque'])
```

```
0.64162034 0.8593748951089336
0.68887085 0.7037187648966691
0.7187992 0.6889763363188675
0.674319 0.7145388288641314
0.6732159 0.8589167538506266
0.74029523 0.8761964461298001
0.59954 0.9422470606237429
0.69299656 1.0
0.68343085 0.7551577195601546
0.7909594 1.0
```

可以看到，通过以上 `Tagger` ，已经将 ARNIQA 与 BRISQUE 分数嵌入到数据集中，我们便可以分析数据集的分布情况。

In [15]:
```
score_arniqa = [d['__tag_arniqa'] for d in tag_dataset]
score_brisque = [d['__tag_brisque'] for d in tag_dataset]
```

In [16]:
```
import scipy
scipy.stats.describe(score_arniqa)
```

Out[16]:
```
DescribeResult(nobs=10, minmax=(0.59954, 0.7909594), mean=0.6904047, varian
ce=0.002739419901950492, skewness=0.23519346591831602, kurtosis=0.011748301
514658444)
```

In [17]:
```
scipy.stats.describe(score_brisque)
```

Out[17]:
```
DescribeResult(nobs=10, minmax=(0.6889763363188675, 1.0), mean=0.8399126805
352927, variance=0.014224137561604368, skewness=0.05486848219025151, kurtos
is=-1.4214439254874505)
```

通过上文提到的 `ensemble` 过滤器，结合不同的权重，可以过滤出不同数量的采样数据集。

`ensemble` 过滤器的接口如下：

```
def ensemble(
    item: T,
    taggers: Sequence[Tagger | str],
    weights: Sequence[float],
    min_score: float = 0.,
    max_score: float = 1.,
    ) -> bool:
```

其中，

- item，表示需要过滤的对象
- taggers，可以是 Tagger 或者 str，表示 item 可供过滤的项
- weights，表示各个 Tagger 的权重
- min_score，过滤的最小分数
- max_score，过滤的最大分数

In [18]:
```
from paddlemix.datacopilot.ops.filter import ensemble
ensemble_fn = partial(ensemble, taggers=[tag_arniqa, tag_brisque], weights=
new_dataset = tag_dataset.filter(ensemble_fn, max_workers=3)
len(new_dataset)
```

```
100%|██████████| 10/10 [00:00<00:00, 6476.69it/s]
```

Out[18]:
```
9
```

In [19]:
```
ensemble_fn = partial(ensemble, taggers=[tag_arniqa, tag_brisque], weights=
new_dataset = tag_dataset.filter(ensemble_fn, max_workers=3)
len(new_dataset)
```

```
100%|██████████| 10/10 [00:00<00:00, 7033.88it/s]
```

Out[19]:
```
8
```

In [20]:
```
ensemble_fn = partial(ensemble, taggers=[tag_arniqa, tag_brisque], weights=
new_dataset = tag_dataset.filter(ensemble_fn, max_workers=3)
len(new_dataset)
```

```
100%|██████████| 10/10 [00:00<00:00, 8298.98it/s]
```

Out[20]:
```
5
```

可以看到，通过设定不同的权重，可以控制采样的数据量大小。基于此可以分析不同质量的数据集对最终模型的训练结果有何影响。

总结

本文介绍了 PaddleMIX 中引入的 Filter 过滤器与 Tagger 标注器的使用方法。重点介绍了 ARNIQA 模型与 BRISQUE 模型的使用。

用户可以通过使用 Filter 过滤掉数据集中低质量的数据，并可以结合 Tagger 与 `ensemble` 过滤器综合考量数据集中的不同参考点。

由此，为提供高质量数据集，并分析数据集对模型训练的结果有何影响提供了参考。

In [ ]:

In [ ]: