

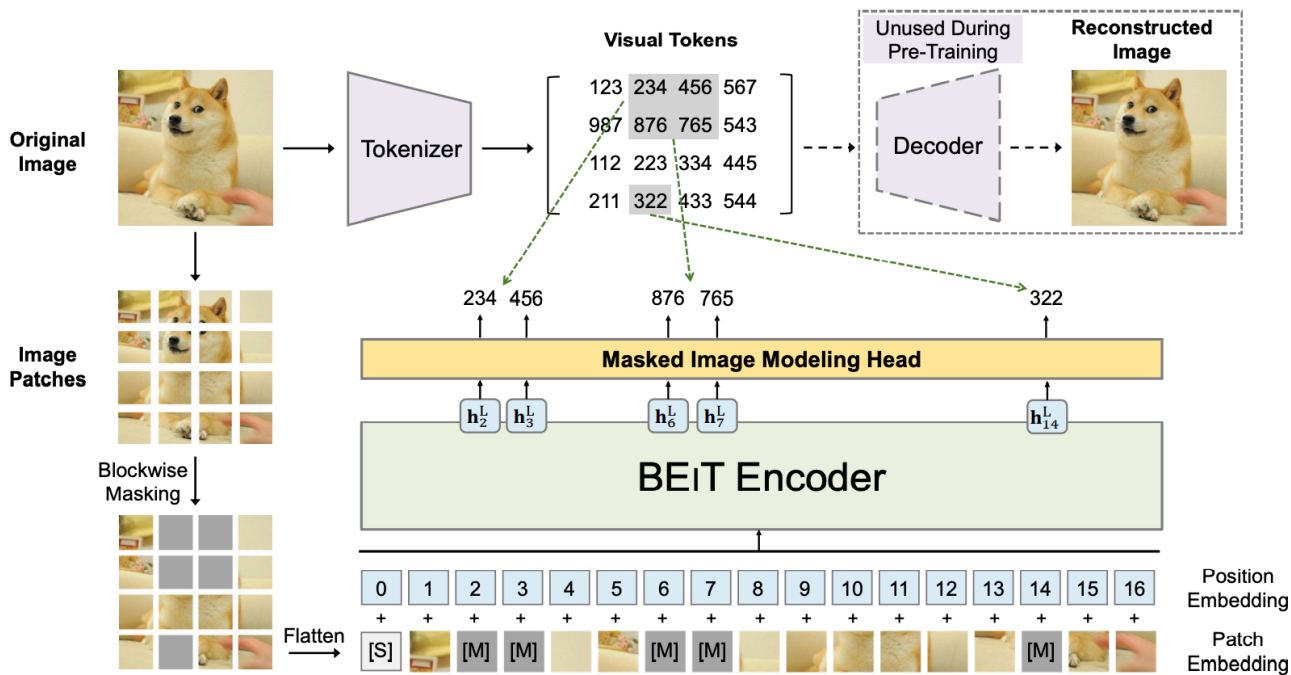
BEiT 模型自验报告

邢朝龙 kaierlong@126.com

1. 模型简介

1.1 网络模型结构简介

本文提出了一种自监督视觉表示模型BEiT。受BERT的启发，作者提出了一个预训练任务，即 **masked image modeling (MIM)**。MIM对每个图像使用两个视图，即图像patch和视觉token。作者将图像分割成一个网格的patches，这些patch是主干Transformer的输入表示。此外，作者将图像“tokenize”为离散视觉token，这是通过离散VAE的潜在代码获得的。在预训练期间，作者随机mask部分图像patch，并将损坏的输入到Transformer。该模型学习恢复原始图像的视觉token，而不是mask patch的原始像素。



1.2 数据集

所用数据集地址：

启智社区：https://git.openi.org.cn/kaierlong/imagenet2012_whole/datasets

中原算力：obs://kaierlong/data/Beit/imagenet2012

使用训练及测试数据集如下：

使用的数据集：ImageNet2012

数据集大小：共1000个类、224*224彩色图像

训练集：共1,281,167张图像

测试集：共50,000张图像

数据格式：JPEG

注：数据在dataset.py中处理。

下载数据集，目录结构如下：

```
└─imagenet
  │└─train          # 训练数据集
  └─val            # 评估数据集
```

1.3 代码提交地址

暂时提交在启智中，私有未开源。

启智仓库地址如下：<https://git.openi.org.cn/kaierlong/Beit-Ascend.git>

中原算力：obs://kaierlong/code/Beit-Ascend

2. 代码目录结构说明

代码目录结构及说明如下：

```
.
├─ README.md      // 说明文档
├─ README_CN.md   // 中文说明文档
├─ image          // 文档图片目录
├─ src
│  │├─ args.py
│  │├─ configs    // 模型参数配置目录
│  │ │├─ beit_base_patch16_224_pt22k_ft22k.yaml // 配置文件
│  │ │└─ parser.py
│  │├─ data       // 数据加载及处理目录
│  │ │├─ __init__.py
│  │ │├─ augment
│  │ │ │├─ __init__.py
│  │ │ │├─ auto_augment.py
│  │ │ │├─ custom_transforms.py
│  │ │ │├─ mixup.py
│  │ │ │└─ random_erasing.py
│  │ │├─ data_utils
│  │ │ │├─ __init__.py
│  │ │ │└─ moxing_adapter.py
│  │ │└─ imagenet.py
│  └─ models      // 模型定义目录
```

```
├── __init__.py
├── beit
│   ├── __init__.py
│   ├── beit.py      // beit定义文件
│   ├── get_beit.py
│   └── misc.py
├── tools      // 相关工具目录
│   ├── __init__.py
│   ├── callback.py
│   ├── cell.py
│   ├── criterion.py
│   ├── get_misc.py
│   ├── optimizer.py
│   └── schedulers.py
├── trainers    // 训练目录
│   ├── __init__.py
│   ├── model_ema.py
│   ├── train_one_step_with_ema.py
│   └── train_one_step_with_scale_and_clip_global_norm.py
├── conv_pt2ckpt.py    // 预训练权重转换
├── eval.py      // 评估文件
└── train.py     // 训练文件
```

3. 自验结果（交付精度规格时需要补齐）

3.1 自验环境

软硬件环境如下：

- 中原算力: Ascend-Powered-Engine | mindspore_1.7.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64
- Ascend: 8*Ascend-910(32GB) | ARM: 192 核 768GB

详细环境配置参见下图:

| | |
|--------|---|
| 算法名称 | algorithm-beit-ascend |
| 预置镜像 | Ascend-Powered-Engine mindspore_1.7.0-cann_5.1.0-py_3.7-euler_2.8.3-aarch64 |
| 代码目录 | /kaierlong/code/Beit-Ascend/ |
| 启动文件 | /kaierlong/code/Beit-Ascend/train.py |
| <hr/> | |
| 计算节点个数 | 1 |
| 规格 | Ascend: 8*Ascend-910(32GB) ARM: 192 核 768GB |
| 作业日志路径 | /kaierlong/output/Beit-Ascend/20221011/logs/ |
| 训练模式 | 普通模式 |

3.2 训练超参数

超参数配置如下：

```
# Architecture
arch: beit_base_patch16_224_pt22k_ft22k

# ===== Dataset ===== #
data_url: ./data/imagenet
set: ImageNet
num_classes: 1000
mix_up: 0.
cutmix: 0.
auto_augment: rand-m9-mstd0.5-inc1
interpolation: bicubic
```

```
re_prob: 0.25
re_mode: pixel
re_count: 1
mixup_prob: 1.0
switch_prob: 0.5
mixup_mode: batch
crop_ratio: 0.875
image_size: 224

# ===== Learning Rate Policy ===== #
optimizer: adamw
lr_scheduler: cosine_lr
base_lr: 0.00002
min_lr: 0.000001
warmup_length: 5
warmup_lr: 0.000001
cool_length: 0
cool_lr: 0.000001
layer_decay: 0.85

# ===== Network training config ===== #
amp_level: 01
keep_bn_fp32: True
beta: [ 0.9, 0.999 ]
is_dynamic_loss_scale: True
use_global_norm: True
clip_global_norm_value: 5.
enable_ema: False
ema_decay: 0.9999
loss_scale: 1024
weight_decay: 0.00000001
momentum: 0.9
label_smoothing: 0.1
epochs: 30
batch_size: 64

# ===== Hardware setup ===== #
num_parallel_workers: 16
device_target: Ascend

# ===== Model config ===== #
drop_path_rate: 0.1
rel_pos_bias: True
abs_pos_emb: False
```

3.3 训练

说明：

因为需要用到预训练模型，需要将pytorch模型进行转换。

提前下载pytorch模型：

- 下载地址：https://conversationhub.blob.core.windows.net/beit-share-public/beit/beit_base_patch16_224_pt22k_ft22k.pth
- 转换命令：

```
# 友情提示需要用到pytorch环境
python3 conv_ptckpt.py --pth_file=beit_base_patch16_224_pt22k_ft22k.pth --
ckpt_file=pretrained_beit.ckpt
```

3.3.1 如何启动训练脚本

训练如何启动：

- 中原算力

模型训练在中原算力完成，完整训练配置如下图所示：

训练输入 ?

data_url

/kaierlong/data/Beit/imagenet2012/

数据集

数据存储位置

--data_url=/home/ma-user/modelarts/inputs/data_url_0

data

pretrained

/kaierlong/data/Beit/pretrained_beit.ckpt

数据集

数据存储位置

--pretrained=/home/ma-user/modelarts/inputs/pretrained_1

ckpt

增加训练输入

训练输出 ?

train_url

/kaierlong/output/Beit-Ascend/20221011/

数据存储位置

--train_url=/home/ma-user/modelarts/outputs/train_url_0

model

增加训练输出

超参

device_num

=

8

device_target

=

Ascend

run_modelarts

=

True

arch

=

beit_base_patch16_224_pt22k_ft22k

增加超参

环境变量

增加环境变量

故障自动重启

- 本地命令

如果需要本地训练，可以使用如下命令：

```
python3 train.py --run_modelarts=True --arch=beit_base_patch16_224_pt22k_ft22k --pretrained=${pretrained} --device_num=8
```

3.3.2 训练精度结果

- 论文精度如下:

| name | initialized checkpoint | resolution | acc@1 | acc@5 | #params | weight | log |
|------------|--|------------|-------|-------|---------|----------------------|----------------------|
| BEiT-base | beit_base_patch16_224_pt22k | 224x224 | 83.7 | 96.6 | 87M | link | link |
| BEiT-base | beit_base_patch16_224_pt22k_ft22k | 224x224 | 85.2 | 97.6 | 87M | link | link |
| BEiT-base | beit_base_patch16_224_pt22k_ft22k | 384x384 | 86.8 | 98.1 | 87M | link | link |
| BEiT-large | beit_large_patch16_224_pt22k | 224x224 | 86.0 | 97.6 | 304M | link | link |
| BEiT-large | beit_large_patch16_224_pt22k_ft22k | 224x224 | 87.4 | 98.3 | 304M | link | link |
| BEiT-large | beit_large_patch16_224_pt22k_ft22k | 384x384 | 88.4 | 98.6 | 305M | link | link |
| BEiT-large | beit_large_patch16_224_pt22k_ft22k | 512x512 | 88.60 | 98.66 | 306M | link | link |

- 复现精度如下:

模型地址（中原算力）：[obs://kaierlong/output/Beit-](#)

[Ascend/20221011/ckpt_best_0000/beit_base_patch16_224_pt22k_ft22k_0000-25_2502.ckpt](#)

| HCS Online | | 中文（简体） | zyrgznjszx-ke... | ? |
|---------------|--|--------|------------------|--------------------------|
| 当前日志文件0.75MB; | | 系统日志 | 请输入关键字查询 | Q As * ^ v ALL |
| 10413 | epoch: 29 step: 2444, loss is 1.6538400650024414 | | | |
| 10414 | epoch: 29 step: 2444, loss is 1.7599453926086426 | | | |
| 10415 | epoch: 29 step: 2444, loss is 1.847732424736023 | | | |
| 10416 | epoch: 29 step: 2444, loss is 1.5448737144470215 | | | |
| 10417 | epoch: 29 step: 2444, loss is 1.503934621810913 | | | |
| 10418 | epoch: 29 step: 2444, loss is 1.5709596872329712 | | | |
| 10419 | epoch: 29 step: 2444, loss is 1.6249059438705444 | | | |
| 10420 | epoch: 29 step: 2444, loss is 1.7422895431518555 | | | |
| 10421 | epoch time: 1308003.184 ms, per step time: 522.783 ms | | | |
| 10422 | epoch time: 1315208.718 ms, per step time: 525.663 ms | | | |
| 10423 | epoch time: 1311663.965 ms, per step time: 524.246 ms | | | |
| 10424 | epoch time: 1310805.306 ms, per step time: 523.903 ms | | | |
| 10425 | epoch time: 1307453.306 ms, per step time: 522.563 ms | | | |
| 10426 | epoch time: 1310666.598 ms, per step time: 523.848 ms | | | |
| 10427 | epoch time: 1316166.271 ms, per step time: 526.046 ms | | | |
| 10428 | epoch time: 1312942.769 ms, per step time: 524.757 ms | | | |
| 10429 | epoch: 0029 device: 0001 acc: 0.8518926056338029, best epoch: 0025, acc is 0.8521927016645326 | | | |
| 10430 | epoch: 0029 device: 0005 acc: 0.8518926056338029, best epoch: 0025, acc is 0.8521927016645326 | | | |
| 10431 | epoch: 0029 device: 0003 acc: 0.8518926056338029, best epoch: 0025, acc is 0.8521927016645326 | | | |
| 10432 | epoch: 0029 device: 0000 acc: 0.8518926056338029, best epoch: 0025, acc is 0.8521927016645326 | | | |
| 10433 | epoch: 0029 device: 0007 acc: 0.8518926056338029, best epoch: 0025, acc is 0.8521927016645326 | | | |
| 10434 | epoch: 0029 device: 0002 acc: 0.8518926056338029, best epoch: 0025, acc is 0.8521927016645326 | | | |
| 10435 | epoch: 0029 device: 0004 acc: 0.8518926056338029, best epoch: 0025, acc is 0.8521927016645326 | | | |
| 10436 | epoch: 0029 device: 0006 acc: 0.8518926056338029, best epoch: 0025, acc is 0.8521927016645326 | | | |
| 10437 | epoch: 30 step: 192, loss is 1.5671995878219604 | | | |
| 10438 | epoch: 30 step: 192, loss is 1.5291666984558105epoch: 30 step: 192, loss is 1.74789297580719 | | | |
| 10439 | | | | |
| 10440 | epoch: 30 step: 192, loss is 1.7532824277877808 | | | |
| 10441 | epoch: 30 step: 192, loss is 1.4707789421081543epoch: 30 step: 192, loss is 2.11814022064209epoch: 30 step: 192, loss is 2.207365036010742 | | | |
| 10442 | | | | |
| 10443 | | | | |
| 10444 | epoch: 30 step: 192, loss is 1.9364129304885864 | | | |
| 10445 | epoch: 30 step: 442, loss is 1.9083691835403442 | | | |
| 10446 | epoch: 30 step: 442, loss is 1.595210313796997epoch: 30 step: 442, loss is 1.619685411453247 | | | |
| 10447 | | | | |

- 精度结果对比

- 论文精度为: 85.2
- 复现精度为: 85.219（最优值）

3.4 模型推理

推理命令如下：

```
python3 eval.py --config=src/configs/beit_base_patch16_224_pt22k_ft22k.yaml --  
pretrained={ckpt_path} --device_id={device_id} --device_target={device_target} --  
data_url={data_url}
```

4. 参考资料

4.1 参考论文

- [BEiT: BERT Pre-Training of Image Transformers](#)

4.2 参考git项目

- [microsoft/unilm/beit](#)

4.3 参考文献

- [《BEiT》-基于图像重建进行预训练！微软提出BEiT，Top-1准确率达86.3%！代码已开源！](#)