甲骨文识别

目录

/oracle-recognition
├─── /data | 数据目录
│   ├─── /raw_data | 原始采样数据（未生成数据集）
│   │   ├─── /oracle-recognition
│   │   └─── /mnist
│   ├─── /source_data | 基础数据集
│   └─── /preprocessed_data | 预处理后数据
├─── /notebook | jupyter notebook 用于实验和可视化
├─── /papers | 部分重要论文资料
└─── /src | 源代码根目录
├─── /models | 包含各类模型函数和工具函数，用于构建模型
│   ├─── __init__.py | 头文件
│   ├─── capsNet.py | Capsule网络
│   ├─── capsNet_distribute.py | Capsule网络（分布式）
│   ├─── capsNet_multi_tasks.py | Capsule网络（分布式、batch_size增强版）
│   ├─── caps_activate_fn.py | capsule的激活函数
│   ├─── capsule_layers.py | 各类capsule模型
│   ├─── get_transfer_learning_codes.py | 直接获得迁移学习模型的bottleneck features
│   ├─── layers.py | 主要的模型架构层
│   ├─── transfer_models.py | 用到的迁移学习网络结构
│   └─── utils.py | 工具函数集合
├─── download_data.py | 下载数据集，包括MNIST和CIFAR
├─── preprocess.py | 数据预处理（包括获取迁移学习的bottleneck features）
├─── main.py | 运行主文件，用于训练模型
├─── test.py | 测试主文件，用于训练测试和评估
├─── fine_tune.py | 迁移学习fine-tuning文件，生成bottleneckfeatures
├─── config.py | 配置文件，包含各类参数和超参数的设置
├─── capsNet_arch.py | 模型结构设置
├─── baseline_config.py | baseline的设置
├─── baseline_arch.py | baseline的模型结构
├─── pipeline.py | pipeline训练，一次性跑多个配置的多个模型
├─── config_pipeline.py | pipeline运行的配置文件

```
├─── /capsulesEM_V1 | MatrixCapsule方法1目录
├─── /capsulesEM_V2 | MatrixCapsule方法2目录
├─── Capsule_Keras.py | Keras版本capsule
├─── capsule_test_Keras.py | Keras版本的运行文件
├─── /data_gen | 问卷调查数据征集相关代码
|    ├─── generate_sheet.py | 生成问卷
|    └─── scan.py | 扫描问卷
├─── imgs_to_black.pyy | 将白底的图转为黑色
├─── clear_all_logs.sh | bash命令文件，清除所有训练记录
├─── remove_all_data.sh | bash命令文件，清除所有预处理的数据
└─── zip_logs.sh | bash命令文件，将所有训练log打包
```

数据准备

1. 下载MNIST或CIFAR数据集

   ```
   python download_data.py
   ```

   然后，输入指令选择下载数据集:

   ```
   ========================================================
   Input [ 1 ] to download the MNIST database.
   Input [ 2 ] to download the CIFAR-10 database.
   Input [ 3 ] to download the MNIST and CIFAR-10 database.
   --------------------------------------------------------
   Input:
   ```

2. 部署甲骨文数据集
   将解压后数据集中的 `raw_data` 文件夹中的所有文件夹，放入 `/data/raw_data` 目录中。

数据预处理

```
python preprocess.py
```

> *运行时可选参数:*
> *–h, ––help show this help message and exit*

> *-b, --baseline Use baseline configurations.*
> *-m, --mnist Preprocess the MNIST database.*
> *-c, --cifar Preprocess the CIFAR-10 database.*
> *-o, --oracle Preprocess the Oracle Radicals database.*
> *-t1, --tl1 Save transfer learning cache data.*
> *-t2, --tl2 Get transfer learning bottleneck features.*
> *-si, --show_img Get transfer learning bottleneck features.*

参数设置在 `config.py` 中：

```python
# Setting test set as validation when preprocessing data
__C.DPP_TEST_AS_VALID = True

# Rate of train-test split
__C.TEST_SIZE = 0.2

# Rate of train-validation split
__C.VALID_SIZE = 0.1

# Resize images
__C.RESIZE_INPUTS = False
# Input size
__C.INPUT_SIZE = (28, 28)

# Resize images
__C.RESIZE_IMAGES = False
# Image size
__C.IMAGE_SIZE = (28, 28)

# Using data augment
__C.USE_DATA_AUG = False
# Parameters for data augment
__C.DATA_AUG_PARAM = dict(
    rotation_range=40,
    width_shift_range=0.4,
    height_shift_range=0.4,
    # shear_range=0.1,
    zoom_range=[1.0, 2.0],
    horizontal_flip=True,
    fill_mode='nearest'
)
```

```
# Keep original images if use data augment
__C.DATA_AUG_KEEP_SOURCE = True
# The max number of images of a class if use data augment
__C.MAX_IMAGE_NUM = 2000
# Change poses of images
__C.CHANGE_DATA_POSE = False

# Oracle Parameters
# Number of radicals to use for training
# Max = 148
__C.NUM_RADICALS = 148

# Preprocessing images of superpositions of multi-objects
# If None, one image only shows one object.
# If n, one image includes a superposition of n objects, the positions of
# those objects are random.
__C.NUM_MULTI_OBJECT = 2
# The number of multi-objects images
__C.NUM_MULTI_IMG = 10000
# If overlap, the multi-objects will be overlapped in a image.
__C.OVERLAP = True
# If Repeat, repetitive labels will appear in a image.
__C.REPEAT = False
# Shift pixels while merging images
__C.SHIFT_PIXELS = 4
```

迁移学习Fine-tune

```
python preprocess.py
```

参数:

```
config:  Configuration
n_output: The output class number
base_model_name: model name for transfer learning
n_use_layers: use the top-n layers. If None, use all
n_freeze_layers: freeze the top-n layers. If None, freeze all
load_pre_model:load a pre-trained model
epochs: epochs for fine-tunning
batch_size: batch size for fine-tunning
```

迁移学习可选以下模型:

> *VGG16: 'vgg16'*
> *VGG19: 'vgg19'*
> *InceptionV3: 'inceptionv3'*
> *ResNet50: 'resnet50'*
> *Xception: 'xception'*

若使用迁移学习，需要先预处理数据用于迁移学习fine-tuning，然后再预处理数据，使用已经训练好的模型生成bottleneck features。

配置模型架构

在 `capsNet_arch.py` 中配置模型架构，方法和Keras类似:
注意相同的模块之间，idx需要不一样，否则无法计算。

示例:

```
def classifier(inputs, cfg, batch_size=None, is_training=None):

  if cfg.DATABASE_NAME == 'radical':
    num_classes = cfg.NUM_RADICALS
  else:
    num_classes = 10

  model = Sequential(inputs)

  # 添加卷积层
  model.add(ConvLayer(
    cfg,
    kernel_size=9,
    stride=1,
    n_kernel=256,
    padding='VALID',
    act_fn='relu',
    idx=0
  ))

  # 添加卷积Capsule
```

```
    model.add(Conv2CapsLayer(
      cfg,
      kernel_size=9,
      stride=2,
      n_kernel=32,
      vec_dim=8,
      padding='VALID',
      batch_size=batch_size
    ))

    # 添加普通全连接Capsule
    model.add(CapsLayer(
      cfg,
      num_caps=num_classes,
      vec_dim=16,
      route_epoch=3,
      batch_size=batch_size,
      idx=0
    ))

    return model.top_layer, model.info
```

模型架构可以使用 `capsule_layers.py` 和 `layers.py` 中预先写好的一些模型，包括:

> *DenseLayer // Single full-connected layer*
> *ConvLayer // Single convolution layer*
> *ConvTLayer // Single transpose convolution layer*
> *MaxPool // Max Pooling layer*
> *AveragePool // Average Pooling layer*
> *BatchNorm // Batch normalization layer*
> *Reshape // Reshape a tenso*
> *CapsLayer // Capsule Layer with dynamic routing*
> *Conv2CapsLayer // Generate a Capsule layer using convolution kernel*
> *MatrixCapsLayer // Matrix capsule layer with EM routing*
> *Dense2CapsLayer // Single full_connected layer*
> *Code2CapsLayer // Generate a Capsule layer densely from bottleneck features*

此外，模型架构中的一些参数可以在 `config.py` 中设置:

```
# ------------------------------------------
# Classification

# Classification loss
# 'margin': margin loss
# 'margin_h': margin loss in Hinton's paper
__C.CLF_LOSS = 'margin_h'

# Parameters of margin loss
# default: {'m_plus': 0.9, 'm_minus': 0.1, 'lambda_': 0.5}
__C.MARGIN_LOSS_PARAMS = {'m_plus': 0.9,
                          'm_minus': 0.1,
                          'lambda_': 0.5}
# default: {'margin': 0.4, 'down_weight': 0.5}
__C.MARGIN_LOSS_H_PARAMS = {'margin': 0.4,
                            'down_weight': 0.5}

# Add epsilon(a very small number) to zeros
__C.EPSILON = 1e-9

# stddev of tf.truncated_normal_initializer()
__C.WEIGHTS_STDDEV = 0.01

# ------------------------------------------
# Optimizer and learning rate decay

# Optimizer
# 'gd': GradientDescentOptimizer()
# 'adam': AdamOptimizer()
# 'momentum': MomentumOptimizer()
__C.OPTIMIZER = 'adam'

# Momentum Optimizer
# Boundaries of learning rate
__C.LR_BOUNDARIES = [82, 123, 300]
# Stage of learning rate
__C.LR_STAGE = [1, 0.1, 0.01, 0.002]
# Momentum parameter of momentum optimizer
__C.MOMENTUM = 0.9

# ------------------------------------------
# Reconstruction
```

```

# Training with reconstruction
__C.WITH_REC = True

# Type of decoder of reconstruction:
# 'fc': full_connected layers
# 'conv': convolution layers
# 'conv_t': transpose convolution layers
__C.DECODER_TYPE = 'fc'

# Reconstruction loss
# 'mse': Mean Square Error
# 'ce' : sigmoid_cross_entropy_with_logits
__C.REC_LOSS = 'ce'

# Scaling for reconstruction loss
__C.REC_LOSS_SCALE = 0.392  # 0.0005*32*32=0.512  # 0.0005*784=0.392

# ----------------------------------------
# Transfer Learning

# Transfer learning mode
# __C.TRANSFER_LEARNING = 'encode'  # None
__C.TRANSFER_LEARNING = None

# Transfer learning model
# 'vgg16', 'vgg19', 'resnet50', 'inceptionv3', 'xception'
__C.TL_MODEL = 'xception'

# Pooling method: 'avg', None
__C.BF_POOLING = None
```

模型训练

```
python main.py -m
```

> *运行时可选参数:*
> *-h, --help show this help message and exit*
> *-g , --gpu Run single-gpu version.Choose the GPU from: [0, 1]*

> *-bs , --batch_size Set batch size.*
> *-tn , --task_number Set task number.*
> *-m, --mgpu Run multi-gpu version.*
> *-t, --mtask Run multi-tasks version.*
> *-b, --baseline Use baseline architecture and configurations.*

训练时注意调整batch_size大小，否则会内存溢出。

训练时的参数在 `config.py` 中配置：

模型训练超参数

```
# Database name
# 'radical': Oracle Radicals
# 'mnist': MNIST
# 'cifar10' CIFAR-10
# __C.DATABASE_NAME = 'radical'
__C.DATABASE_NAME = 'mnist'
# __C.DATABASE_MODE = 'small_no_pool_56_56'
# __C.DATABASE_MODE = 'small'
__C.DATABASE_MODE = None

# Training version
# Set None to auto generate version
__C.VERSION = None

# Learning rate
__C.LEARNING_RATE = 0.001

# Learning rate with exponential decay
# Use learning rate decay
__C.LR_DECAY = True
# Decay steps
__C.LR_DECAY_STEPS = 2000
# Exponential decay rate
__C.LR_DECAY_RATE = 0.96

# Epochs
__C.EPOCHS = 20

```

```
# Batch size
__C.BATCH_SIZE = 512
```

训练过程流程和显示信息设置

```
# Display step
# Set None to not display details
__C.DISPLAY_STEP = None  # batches

# Save summary step
# Set None to not save summaries
__C.SAVE_LOG_STEP = 100  # batches

# Save reconstructed images
# Set None to not save images
__C.SAVE_IMAGE_STEP = 100  # batches

# Maximum images number in a col
__C.MAX_IMAGE_IN_COL = 10

# Calculate train loss and valid loss using full data set
# 'per_epoch': evaluate on full set when n epochs finished
# 'per_batch': evaluate on full set when n batches finished
__C.FULL_SET_EVAL_MODE = 'per_epoch'
# None: not evaluate
__C.FULL_SET_EVAL_STEP = 1

# Save models
# 'per_epoch': save models when n epochs finished
# 'per_batch': save models when n batches finished
# __C.SAVE_MODEL_MODE = None
__C.SAVE_MODEL_MODE = 'per_epoch'
# None: not save models
__C.SAVE_MODEL_STEP = 5
# Maximum number of recent checkpoints to keep.
__C.MAX_TO_KEEP_CKP = 3

# Calculate the train loss of full data set, which may take lots of time.
__C.EVAL_WITH_FULL_TRAIN_SET = False

# Show details of training progress
```

```
__C.SHOW_TRAINING_DETAILS = False

# ----------------------------------------
# Test
# 'after_training': evaluate after all training finished
# 'per_epoch': evaluate when a epoch finished
# None: Do not test

# Evaluate on single-object test set
__C.TEST_SO_MODE = 'per_epoch'

# Evaluate on multi-objects test set
__C.TEST_MO_MODE = 'per_epoch'

# Evaluate on Oracles test set
__C.TEST_ORACLE_MODE = 'per_epoch'
```

多显卡分布式计算相关设置

```
# Save trainable variables on CPU
__C.VAR_ON_CPU = True

# Number of GPUs
__C.GPU_NUMBER = 2

# Number of tasks
__C.TASK_NUMBER = 4

# The decay to use for the moving average.
# If None, not use
__C.MOVING_AVERAGE_DECAY = 0.9999
```

模型测试和评估

实际上，模型在训练结束后会根据设置自动进行计算和评估，但是也可以通过 `test.py` 自行测试，但是要注意读取的模型位置和模型编号。

```
python test.py
```

> *运行时可选参数:*
> *-h, --help show this help message and exit*
> *-b, --baseline Use baseline configurations.*
> *-mo, --multi_obj Test multi-objects detection.*
> *-m, --mgpu Test multi-gpu version.*
> *-o, --oracle Test oracles detection.*

模型测试相关参数在 `config.py` 中设置，这些设置也会影响训练过程后的评估。

```
# Testing version name
__C.TEST_VERSION = __C.VERSION

# Testing checkpoint index
# If None, load the latest checkpoint.
__C.TEST_CKP_IDX = None

# Testing with reconstruction
__C.TEST_WITH_REC = True

# Saving testing reconstruction images
# If None, do not save images.
__C.TEST_SAVE_IMAGE_STEP = 5  # batches

# Batch size of testing
# should be same as training batch_size
__C.TEST_BATCH_SIZE = __C.BATCH_SIZE

# Top_N precision and accuracy
# If None, do not calculate Top_N.
__C.TOP_N_LIST = [5, 10, 20]

# ----------------------------------------
# Multi-objects detection

# Label for generating reconstruction images
# 'pred': Use predicted y
# 'real': Use real labels y
__C.LABEL_FOR_TEST = 'pred'  # 'real'

# Mode of prediction for multi-objects detection
# 'top_n': sort vectors, select longest n classes as y
```

```
# 'length_rate': using length rate of the longest vector class as threshold
__C.MOD_PRED_MODE = 'top_n'  # 'length_rate'

# Max number of prediction y
__C.MOD_PRED_MAX_NUM = 2

# Threshold for 'length_rate' mode
__C.MOD_PRED_THRESHOLD = 0.5

# Save test prediction vectors
__C.SAVE_TEST_PRED = True
```

Pipeline训练

Pipeline训练主要是用于长时间的放置训练，可以一次性跑多个模型，方法也很简单。

```
python pipeline.py
```

然后，输入指令选择运行模式:

```
============================================================
Input [ 1 ] to run normal version.
Input [ 2 ] to run multi-gpu version.
------------------------------------------------------------
Input:
```

将要修改的参数放在 `config_pipeline.py` 的结尾就可以了。

```
# get config by: from distribute_config import config
config = __C

__C.WITH_REC = False
__C.VERSION = _auto_version(__C)
cfg_1 = copy(__C)

__C.WITH_REC = True
__C.VERSION = _auto_version(__C)
cfg_2 = copy(__C)

```

```
__C.REC_LOSS = 'ce'
__C.VERSION = _auto_version(__C)
cfg_3 = copy(__C)

__C.DECODER_TYPE = 'conv'
__C.REC_LOSS = 'mse'
__C.VERSION = _auto_version(__C)
cfg_4 = copy(__C)

__C.REC_LOSS = 'ce'
__C.VERSION = _auto_version(__C)
cfg_5 = copy(__C)

__C.DECODER_TYPE = 'conv_t'
__C.REC_LOSS = 'mse'
__C.VERSION = _auto_version(__C)
cfg_6 = copy(__C)

__C.REC_LOSS = 'ce'
__C.VERSION = _auto_version(__C)
cfg_7 = copy(__C)
```

其他设置

一些目录设置在 `config.py` 的结尾处。

```
# Source data directory path
__C.SOURCE_DATA_PATH = '../data/source_data'

# Preprocessed data path
__C.DPP_DATA_PATH = '../data/preprocessed_data'

# Oracle labels path
__C.ORACLE_LABEL_PATH = __C.SOURCE_DATA_PATH + '/recognized_oracles_labels.csv'

# Path for saving logs
__C.TRAIN_LOG_PATH = '../train_logs'

# Path for saving summaries
__C.SUMMARY_PATH = '../tf_logs'
```

```
16
17  # Path for saving models
18  __C.CHECKPOINT_PATH = '../checkpoints'
19
20  # Path for saving testing logs
    __C.TEST_LOG_PATH = '../test_logs'
```