# CustomKing User Guide

## 1. How to customize data?

**Step1:**

Customizing data needs to create a new data loading file (assumed to be named Load_data.py) under the customKing/data/datasets folder or its subfolders and write it according to the following code structure, where the Load_data and load_data_fun need to be named by the user. The pass part needs to be written by the user.

```
from customKing.data import DatasetCatalog

def load_data_fun(name,root):
        if name == "load_data_fun_train":
                pass #To get dateset
        elif name == "load_data_fun_valid":
                pass #To get dateset
        elif name == "load_data_fun_train_and_valid":
                pass #To get dateset
        elif name == "load_data_fun_test":
                pass #To get dateset
        elif name=="load_data_fun_train_and_valid_and_test":
                pass #To get dateset
        return dataset

def register_load_data_fun(name,root):
        DatasetCatalog.register(name, lambda: load_data_fun(name,root))
```

**Step2:**

Add the following code in customKing/data/datasets/builtin.py:

```
from Load_data import register_load_data_fun
import os

def register_all_load_data_fun(root):
        names = ["load_data_fun_train",
        "load_data_fun_valid",
        "load_data_fun_train_and_valid",
        "load_data_fun_test",
        "load_data_fun_train_and_valid_and_test"]
        for name in names:
                register_load_data_fun(name,root)

if __name__.endswith(".builtin"):
        # Assume pre-defined datasets live in `./datasets`.
        _root = os.path.expanduser("datasets")
        register_all_load_data_fun(_root)
```

## 2. How to write models?

**Step1:**

Writing model needs to create a new building model file (assumed to be named New_model.py) under the customKing/modeling/meta_arch folder or its subfolders and write it according to the following code structure, where the New_model and new_model need to be named by the user. The pass part needs to be written by the user. The parameters that need to be passed in in the new model are all stored in cfg. For the structure of cfg, see the "How to set configuration parameters?" section.

```
from ..build import META_ARCH_REGISTRY

@META_ARCH_REGISTRY.register()
def new_model(cfg)
    pass #to get new model,
    return model
```

**Step2:**

Add the following code in customKing/modeling/meta_arch/__init__.py:

```
from New_model import new_model
```

## 3. How to set configuration parameters?

All configuration parameters of the entire framework are stored in customKing/config/defaults.py, which uses a tree structure for parameter records. The root node of the whole tree is _C = CN(). To define a parent node, you need to set _C.parent_node = CN(), where parent_node is named by the user. All configuration parameters are set at the leaf nodes. Below is an example.

```
from .config import CfgNode as CN

# ---------------------------------------------------------------------------- #
# Config definition
# ---------------------------------------------------------------------------- #

_C = CN()
# --------------------------------------------------------------------------- #
# Classification model config
# --------------------------------------------------------------------------- #
_C.DATASETS = CN()
_C.DATASETS.TRAIN = "Cifar10_train"      #train dataset
_C.DATASETS.VALID = "Cifar10_valid"      #valid dataset
_C.DATASETS.TEST = "Cifar10_test"        #test dataset

_C.MODEL = CN()
_C.MODEL.META_ARCHITECTURE = "Resnet20"    #select classification model
_C.MODEL.OUTPUT_NUM_ClASSES = 10      #set class num
_C.MODEL.INPUT_IMAGESIZE = (32,32)        #set image size
_C.MODEL.DEVICE = "cuda:0"          #select device
_C.MODEL.JUST_EVAL = False
_C.MODEL.PRE_WEIGHT = False
_C.MODEL.OUTPUT_DIR = "output/"+_C.DATASETS.TRAIN[:-6] \
                                    +"/"+_C.MODEL.META_ARCHITECTURE+"/metric.json"
_C.MODEL.PREWEIGHT = r" "        #The path of saving the pretrain weight

_C.SOLVER = CN()
_C.SOLVER.OPTIMIZER = "SGD"        #select optimizer，see：customKing\solver\build.py
_C.SOLVER.BATCH_SIZE = 128        #Set batch_size
_C.SOLVER.SHUFFLE = True
_C.SOLVER.NUM_WORKERS = 8        #the num workers of the Dataloader
_C.SOLVER.IS_PARALLEL = False      #Whether to use multiple GPUs for training

_C.SOLVER.LR_SCHEDULER_NAME = "Step_Decay"
_C.SOLVER.START_ITER = 0
_C.SOLVER.MAX_EPOCH = 200
_C.SOLVER.MAX_ITER = 64000
_C.SOLVER.BASE_LR = 0.1
_C.SOLVER.MOMENTUM = 0.9
_C.SOLVER.NESTEROV = False
_C.SOLVER.WEIGHT_DECAY = 0.0001
_C.SOLVER.GAMMA = 0.1
_C.SOLVER.STEPS = (32000,48000)
_C.SOLVER.CLR_STEPS = 2000          #if using CLR lr_scheduler, the config need to set.
_C.SOLVER.WARMUP_FACTOR = 1.0 / 1000
_C.SOLVER.WARMUP_ITERS = 1000
_C.SOLVER.WARMUP_METHOD = "linear"
```

The method to obtain all configuration parameters is very simple, just use the following code.

```
from customKing.config.config import get_cfg
cfg = get_cfg()
Train_data_name = cfg.DATASETS.TRAIN        #Get the training dataset name
```

## 4. How to build custom tasks?

CustomKing has a built-in classification task. See tools/Image_Classification/main.py. Other tasks can be written according to this classification task.