

```
path:
windows: D:/workspace/art
linux: /home/jzm/workspace/final
```

以 windows 为例，路径中包含如下的文件夹

```
D:.\
├─data_h5
├─data_info_mat
├─doc
├─matlab
│   └─rus
├─net
├─nninfo
├─pic
│   └─protocol
│       ├──protocol_test_para1
│       ├──protocol_test_para2
│       ├──protocol_test_para4
│       └─protocol_training_para1
├─protocol_training_para2
├─res
└─python
└─rus
└─__pycache__
```

data_h5: 存放 .hdf5 格式的数据集文件，数据集根据 nninfo 中的部分图像调整参数及 pic 中的原始图片数据得到。

data_info_mat: 存放 matlab 生成的原始信号参数。

doc: 文档说明。

matlab: matlab 代码，其中 rus 文件夹存放废弃代码。

net: 存放训练好的网络，及测试结果的混淆矩阵截图。

nninfo: 存放网络或训练参数的 .pkl 文件。

pic: 存放 matlab 产生的原始图片。

python: python 代码，其中 rus 文件夹存放废弃代码。

Matlab 相关代码

complex_exponential_wave.m 复载波信号基本类

fh.m 用于产生跳频信号类

generate_pic.m 保存时频图用的函数，适用于单一信道，或多信道单协议的情况

generate_pic_mul.m 保存时频图用的函数，适用于多信道多协议

get_files.m 获取指定路径下的文件名的函数

get_pic.m 生成时频图图片用的函数，调用即可在默认文加下生成 .jpg 格式图片

link16.m 跳频频率集产生类

msk_modulation.m MSK 调制类
para_est.m 参数估计
pro_src_data.m 根据协议生成特定的信号
psk_modulation.m PSK 调制类
qam_modulation.m QAM 调制类
rx_signal.m 接收信号类
src_para.m 产生基本信号参数
t.m 测试用
test.m 测试用
tfdec.m 参数估计类
timeslot_est.m 时隙估计类

产生不同协议的时频图方法:

简略:使用 src_para.m 保存 .mat 文件, 在使用 get_pic 即可。具体细节如下:

记得修改 src_para.m 中的 index 后, 在 get_pic 中也要修改相应的 file_number, 如果不知道文件编号, 执行 get_files 获取。

参数设置

在 src_para.m 文件中设置基本的参数, 文件中的主要参数如下:

save2mat 是否保存为 .mat 文件
index 若保存文件, index 为文件的编号, 文件命名格式: {data_type}para{index}.mat
pic_number 设置需要产生的图片个数, 在单一协议下, 该参数表示一种协议产生的图片数, 总图片数等于 $4 * \text{pic_number}$ 。在多协议的条件下, pic_number 为最终生成的图片数量。
multi 多协议标志 (0:单协议 1:多协议) 单协议或多协议指的是在一张时频图中出现的协议数量
freq_num 频点数量 (信道数量)
rand_select 是否随机生成频率
data_type "test" or "training"

文件默认存放的路径为 data_info_mat 文件夹所在路径, 已设置双系统下的默认路径

protocol_type 默认 4 种
package_len 协议中包的长度
mod_para 每个信道中用户的参数
fs 采样率
sample_length 样本总长度
slot_len 时隙长度
slot_info 时隙信息, 对于 aloha 和 csma 无效
channel 信道信息, 字符串, 指定为高斯信道或衰落信道
snr 为高斯信道时指定信噪比

保存文件时, 将 save2mat 置为 1, 生成不同参数的文件时, 记得修改 index 的数值

单协议参数

单协议指每张时频图中只存在一种协议，频点个数可以是一个，也可以是多个。

单一信道(频点)

```
multi = 0、freq_num = 1
```

rand_select 选择是否随机产生频率，为 0 时可设置频率值，(要修改在源代码 51 行处修改)，为 1 时从 link 16 的 51 个频点值中随机抽取。

多信道(频点)

```
multi = 1、freq_num
```

rand_select 选择是否随机产生频率，为 0 时可设置频率值，(要修改在源代码 51 行处修改)，为 1 时从 link 16 的 51 个频点值中随机抽取。



保存图片命名格式： {protocol_name}_{pic_number}.jpg

多协议参数

固定频点

```
multi = 1、freq_num、rand_select = 0
```

可变频点

```
multi = 1、freq_num、rand_select = 1
```



二者的区别在于：固定频点的所有图片使用的频点相同，可变频点则可能使用不同的频点。

注意命名格式：{protocol_name1}-{protocol_name2}_{pic_number}.jpg

每张图片包含多个协议，协议按照在图片中的位置由上到下依次命名，中间用 '-' 隔开。

全过程的采样率涉及带通采样，选择不同的采样率会得到不同的等效频率，而选择的 610 MHz 采样率，得到的等效频率与原频率相比正好是相反的关系，即原频率越高，采样后的频率越低，具体细节可见相关文档，link16 类中包含等效频率的具体数值。

Python 相关代码

[dataset.py](#) 数据集制作及导入

[ds.py](#) 数据集加载及查看

[gantest.py](#) 源于网络

[net.py](#) 网络结构、训练、测试

[nn cal.py](#) 测试网络结构及参数是否正确

[nnpar.py](#) 数据集制作中的图片处理参数设置、网络设置、训练参数测试

[par.py](#) 多模块共用参数存放处，命令行参数设置

[path.py](#) 多模块共用路径存放处

[pic.py](#) 图片处理函数，将图片有信号部分切割并移动到中心

[plotcm.py](#) 混淆矩阵绘制相关

[t.py](#) 测试

[test.py](#) 测试

[tt.py](#) 测试

[ttest.py](#) 主要内容

[plotcm.py](#) 混淆矩阵绘制相关

pic.py

实现图片切割，对于只有一个频点的时频图，由于其频点值可能发生改变，导致信号出现的位置不固定，而使用某一频点的数据训练网络后，使用另一频点的数据进行验证，其效果一般不理想，推测可能与时频图中信号出现的位置有关，为了消除这一影响，决定对原始时频图进行一定的处理，将所有时频图中出现的信号移动到图片的中央或近中央附近，再进行后续的训练。对于只有一个频点的时频图，找出其包含信号的部分，取含有信号的部分及其上下 `exten(pic.py` 中的变量) 附近的图片移动至图片中央；对于有多个频点的时频图，则分别取出每一段含有信号的频点，再分别移动到图片中央，注意此时的图片数量为频点的数量，因此此时的数据集的大小将是频点数量与图片数量的乘积。训练时均采用只含一个频点的图片进行训练。

path.py

存放一些文件夹的地址。

`get_dataset_path(path, **kwargs)` 返回指定路径中非空文件夹的名称，用于根据图片制作数据集。最初时使用，现可能使用较少或者不推荐使用。

`get_file(path)` 函数返回指定路径中的文件名

par.py

内容较少，主要有一个命令行参数内容的设置

nnpar.py

定义一些参数，如 `pic_size` 定义处理后的图片尺寸，`pic_list` 定义图片处理的具体方法，`pic_enhance_list` 为空时不进行数据增强，需要图片增强时使用上一句被注释的语句，数据增强内容为简单的水平翻转，只对训练集使用数据增强。

`batch_size`、`epoch`、`learning_rate` 为训练时的参数

`nn_list` 用于定义网络结构，网络结构需要先在 `nncal.py` 中进行验证

最后将这些参数打包定义一个 `trainpar` 类

命令行参数

`--show` 显示部分参数

`--save` 保存为 `.pkl` 文件 需要保存为 `.pkl` 文件时一定要有

`--id` 用于指定 `.pkl` 文件的编号，缺省时默认为零 `.pkl` 文件的命名格式：`par_{index}.pkl`

保存为 `.pkl` 文件后会有提示信息

使用：

```
python nnpar.py --save --id 2
```

nncal.py

用于网络结构正确性的验证，也可用于 `.pkl` 文件中参数正确性的验证。网络结构的验证运用了 `torchsummary` 模块中的 `summary`，具体使用方法可参考代码，也可参考相关资料。

命令行参数

--show 显示参数(基本源于 [nnpar.py](#) 中的参数)

--id 用于指定 .pkl 文件的编号, 缺省时默认为零

一般 [nnpar.py](#) 和 [nncal.py](#) 结合使用, 先使用 [nnpar.py](#) 得到 .pkl 文件, 再验证网络的正确性。网络可在别的文件中决定, 经过验证后复制到 [nnpar.py](#) 中, 网络定义的方法参考 [nnpar.py](#), 网络验证的方法参考 [nncal.py](#)

使用:

```
python nncal.py --id 2 --show
```

net.py

trainpar 类, 训练参数类

residual_block 类和 resnet 类 用于搭建残差网络

```
model = net.resnet(net.residual_block, [2,2,2,2])    ### from nnpar.py
```

neuralnetwork 类, 根据列表构建网络

```
#### from nnpar.py
nn_list = [ ('conv1', nn.Conv2d(3, 10, kernel_size=5)),
            ('max_pool1', nn.MaxPool2d(kernel_size=2)),
            ('relu1', nn.ReLU(inplace=True)),
            ('conv2', nn.Conv2d(10, 32, kernel_size=5)),
            ('max_pool2', nn.MaxPool2d(kernel_size=2)),
            ('relu2', nn.ReLU(inplace=True)),
            # ('dropout1', nn.Dropout2d()),
            ('flatten1', nn.Flatten(start_dim=1)),
            ('affine1', nn.Linear(32 * 45 * 45, 50)),
            ('affine2', nn.Linear(50, 10)),
            ('affine3', nn.Linear(10, 4)),
            ]
```

```
nn_list = OrderedDict(nn_list)
model = net.neuralnetwork(nn_list)
```

train 和 test 函数分别定义训练及测试的一些操作

ds.py

根据文件名加载 .hdf5 格式的数据集, 如果没有该文件, 则先创建 .hdf5 格式的数据集文件再加载。

dataset.py

根据 .jpg 格式的图片, 制作 .hdf5 格式的数据集, 其中有两个版本的函数, 一个用于单一频点数据集的制作(主要是训练集), 另一个带 _mul 后缀的函数制作多频点数据集, 主要用于测试集的制作。

另有根据 .jpg 图片直接制作数据集的函数，但该方法会将占用大量内存(跟制作 .hdf5 文件时类似)，不建议大量数据时使用，现已废弃，可在 `rus/dataset_.py` 中找到相关代码。

ttest.py

主要使用的是 `ttest.py` 这个模块，用于训练或测试。使用时一定要输入 `--test` 或 `--train` 区分用于测试还是训练，否则无法完成。

命令行参数(定义在 `par.py` 中)

`--test` 测试时输入

`--train` 训练时输入

`--create` 强制创建 .hdf5 数据集文件(若已存在 .hdf5 文件，删除后重建)

`--te` 指定测试数据集编号，缺省值为 1

`--tr` 指定训练数据集编号，缺省值为 1

`--npa` 指定参数 .pkl 文件编号缺省值为 0

`--mul` 对于多频点的数据，需加上

`--test` 会确定 `data_type` 为 `test`

`--train` 会确定 `data_type` 为 `train`

缺少上述两参数之一，则 `data_type` 未定义

.hdf5 文件的命名格式: `protocol_{data_type}para{para_index}_nnpar{nnpar_index}.hdf5`

`data_type` 表示数据的类型， `test` or `train`

`para_index` 表示 .mat 文件的编号

`nnpar_index` 表示 .pkl 文件的编号

.mat 文件的命名格式: `{data_type}para{para_index}.mat`

.pkl 文件的命名格式: `par{nnpar_index}.pkl`

注意训练与测试两个过程分离，训练后不会进行测试

训练

根据 `/pic/protocol/protocol_training_para1` 进行训练

```
python ttest.py --train --tr 1
## if exist data_h5/protocol_test_para1_nnpar1.hdf5 load it and training
```

```
python ttest.py --train --tr 1 --create
## remove data_h5/protocol_test_para1_nnpar1.hdf5 (if exist)
## create data_h5/protocol_test_para1_nnpar1.hdf5
## load it and training
```

测试

根据 `/pic/protocol/protocol_test_para4` 进行测试，测试的模型由 `/pic/protocol/protocol_training_para1` 训练得到

```
python ttest.py --test --te 4 --tr 1 (--create)
```