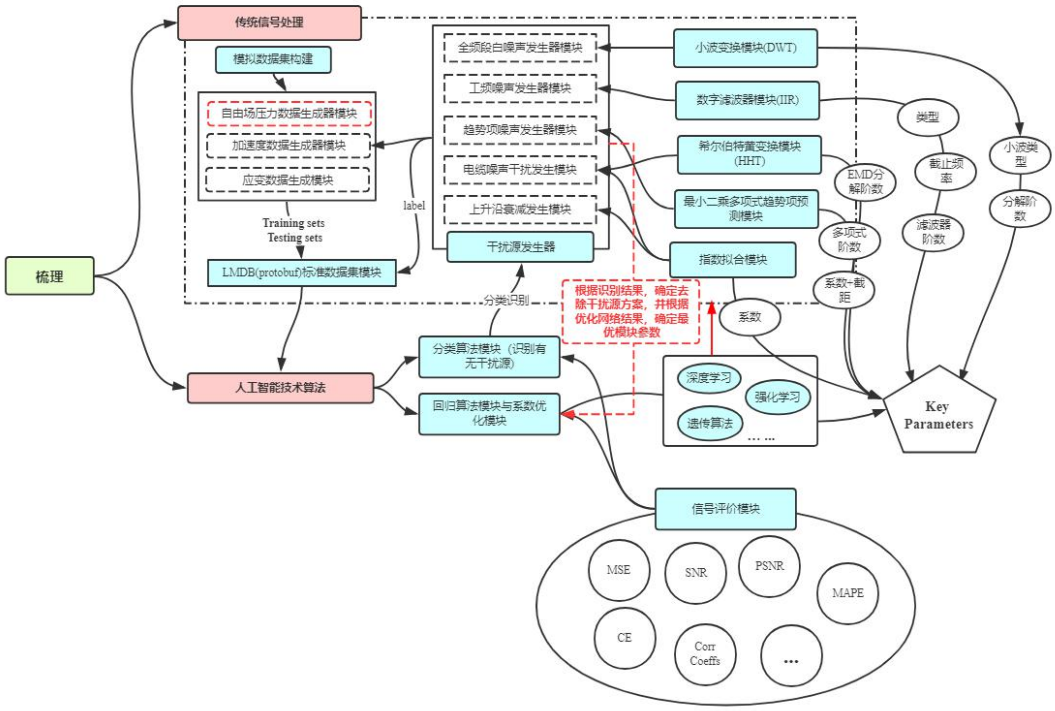
(Demo)自由场压力数据集的 4 阶趋势项去除

✓ 总体模块架构与 demo 流程



➢ 训练数据生成

1) 包含模块：自由场压力数据生成器模块、加速度数据生成器模块、应变数据生成器模块、LMDB(Protobuf)标准数据集模块、各干扰源发生器模块(工频、白噪、趋势项等)、 各干扰源处理算法模块(小波变换模块 DWT、数字滤波模块 IIR、希尔伯特黄变换 HHT 等)；

2) 功能：

• 用于干扰源识别任务中，基于模拟或者真实的小批干净源数据，随机叠加不同类型的噪声数据，并统一包含噪声类型的数据标签，统一批量打包成标准数据集；

• 用于信号处理算法的参数优选中，基于模拟或者真实的小批干净源数据，叠加不同类型的噪声数据，并将处理该类型噪声数据的处理算法的专家先验参数作为数据标签，统一打包成标准数据集；

3) 输入：相应的数据集(以文件形式或以一维数组数据流形式)、需要加入的噪声种类、处理对应噪声的处理算法的专家先验参数；

4) 输出：以输入为基础，可用于训练算法模型的带有标签的标准数据集以及可用于进行测试的带有标签的标准数据集。

➢ 干扰源识别

1) 包含模块: 分类算法模块、信号评价模块；

2) 功能：对于数据集中包含的噪声类型进行分析于识别，后续方便工作人员选择去除

该干扰源的处理算法；

3) 输入：4.3.2 中输出的带有标签的用于干扰源任务类型的标准数据集；

4) 输出：信号中包含的整体信号干扰类型和瞬时信号干扰类型；前者用固定三个数字片段表示，第一个数字表示是否包含电磁干扰，第二个数字表示是否包含零飘项，第三个数字表示是否包含趋势项（如“001”表示信号中没有电磁干扰、没有零飘项、有趋势项）。后者未一维数组数据流形式，表示各个短时片段中是否包含电缆噪声干扰等；

5）实现逻辑：内部先对信号进行分片提特征，然后调用 pytorch 或 tensorflow 组件进行深度神经网络模型运算，用预训练的干扰源识别模型判断各个片段是否包含干扰源，最后时刻提取固定维度特征判断是否包含不同类型的整体时域干扰源；

- **处理算法参数的智能选取**

1) 包含模块：回归算法模块与系数优化模块、信号评价模块；

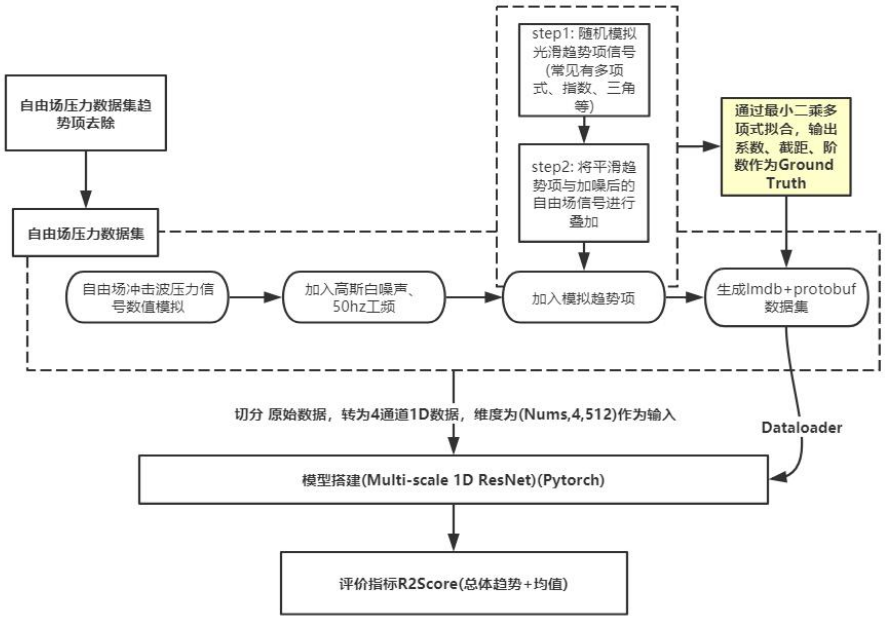
2) 功能：对标准数据集中信号的包含的某个干扰源类型的指定的处理算法，预测对该处理算法的性能起到决定作用的关键参数；

3) 输入：4.3.2 中输出的带有标签的用于信号处理算法的参数优选的标准数据集；

4) 输出：指定处理算法的关键参数；

5) 实现逻辑：这里以自由场压力数据趋势项去除关键参数(阶数、系数、截距)为例进行说明：

- 基于**深度学习**的关键参数回归： 首先，对带有专家先验标签的自由场压力数据集进行训练，从信号评价模块中选择相应评价指标用作分析训练模型的性能，直到模型达到性能阈值。利用该模型，可对自由场压力信号的趋势项的关键参数进行预测；
- 基于**强化学习**的关键参数预测：首先，将关键参数设定为强化学习架构中的 Action space，以数据集中的信号样本作为 State space，从信号评价模块中选择相应模块作为每个时间步采取 action 后，对于下一状态进行评价的 Reward function，每次将当前状态、采取的动作、奖励信息、下一动作作为一个 Transition 存入经验池中；每个时间步都会通过从经验池中抽取小批量的 Transition 进行重放并进行训练，通过反向传播更新动作价值网络参数，总体使每次选择 Action 逼近可以获得最大动作价值的最优动作，达到对于参数优选的目的。



STEP1 自由场冲击波压力信号数值模拟

根据自由场冲击波压力模拟算法，构建纯净的压力数据，并添加全频段高斯白噪声、48~52Hz 工频噪声，并叠加模拟拟合的趋势项，效果如下：

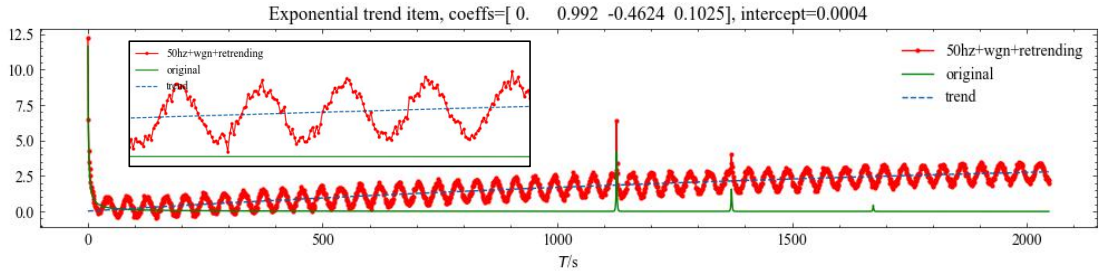


图 1 纯净模拟信号与加噪后的信号 demo

STEP2 生成 lmdb+Protobuf 数据集，方便后期调用

批量生成自由场压力加噪后的数据集，并将不同的 4 阶的系数与截距作为标签，存为.csv 文件，接着生成.mdb 的标准文件，使用 Protobuf 的 lmdb 可以有效的方便 Pytorch 读取进行训练。

和 Json、xml 一样，protobuf 是一种网络通信和通用数据交换等应用场景中的一种技术，其在效率和兼容性等方面非常的出色。

是一种语言无关、平台无关、可扩展的序列化结构数据的方法，它可用于（数据）通信协议、数据存储等。

Protocol Buffers 是一种灵活，高效，自动化机制的结构数据序列化方法－可类比 XML，其特点如下：

- ✓ 语言无关、平台无关。即 ProtoBuf 支持 Java、C++、Python 等多种语言，支持多个平台
- ✓ 高效。即比 XML 更小（3～10 倍）、更快（20～100 倍）、更为简单
- ✓ 扩展性、兼容性好。你可以更新数据结构，而不影响和破坏原有的旧程序

你可以定义数据的结构，然后使用特殊生成的源代码轻松的在各种数据流中使用各种语言进行编写和读取结构数据。你甚至可以更新数据结构，而不破坏由旧数据结构编译的已部署程序。

图 2 生成的.mdb 数据文件

STEP3 Pytorch Dataloader 并进行预处理

读取.mdb 文件，设立 batch_size，并对原始数据进行切分，切分成 4 通道数据，数据 Dim 为(N,4,512)；

STEP4 基于 Pytorch 搭建基于 ResNet1D 的深度神经网络

训练过程如下，采用预测多项式的系数与截距与 Label 的系数与截距的 MSE 作为 Training Loss Function，再以 R2 Score 来作为 Validation 的指标，训练过程如下：

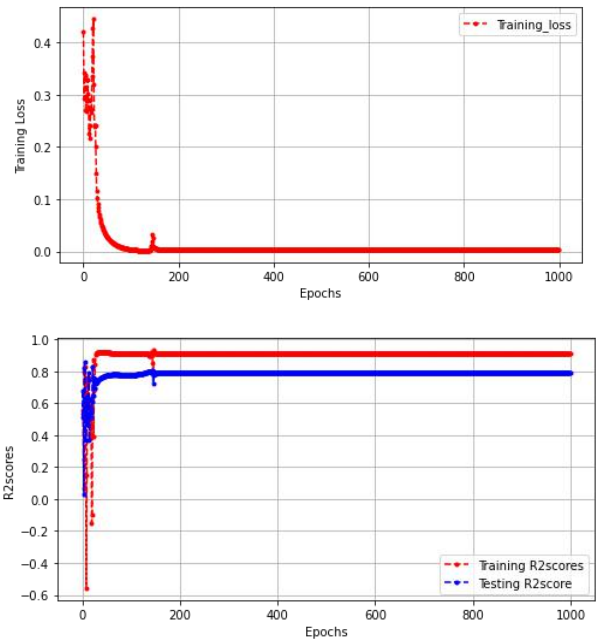


图 3 训练过程可视化

STEP5 基于测试集中随机抽取数据做 inference

抽取了一组比较好的推理效果，可以发现，实现了趋势项的去除，后续还需要继续优化。

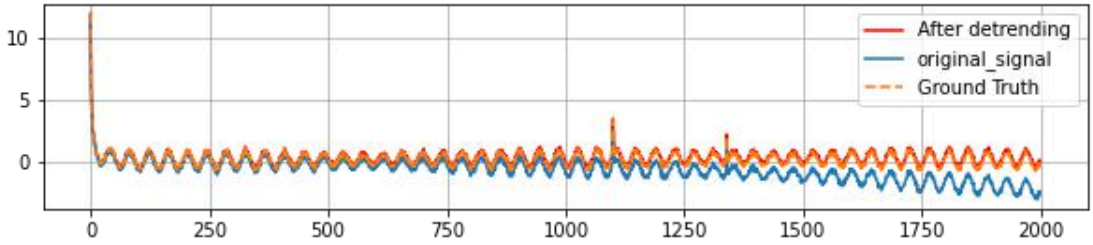


图 4 推理测试用例