This file is written for people who want to utilize this code for other research purpose in Pytorch framework, enable them to have an overview of the source code and can later on easily get familiar with the source code.

- Introduction:

This code is a reimplementation of the paper [Efficient Amortised Bayesian Inference for Hierarchical and Nonlinear Dynamical Systems](https://arxiv.org/abs/1905.12090) in Pytorch framework (the original framework is in tensorflow)

- Brief explanation of the paper:

VI-HDS is a a flexible, scalable Bayesian inference framework for nonlinear dynamical systems characterised by distinct and hierarchical variability at the individual, group, and population levels.
We cast parameter inference as stochastic optimisation of an end-to-end differentiable, block-conditional variational autoencoder.
We specify the dynamics of the data-generating process as an ordinary differential equation (ODE) such that both the ODE and its solver are fully differentiable.
This model class is highly flexible: the ODE right-hand sides can be a mixture of user-prescribed or "white-box" sub-components and neural network or "black-box" sub-components.
Using stochastic optimisation, our amortised inference algorithm could seamlessly scale up to massive data collection pipelines (common in labs with robotic automation).

(Here I reimplement the code in Pytorch to train the hierarchical blackbox model (the most powerful and flexible model in the paper))

- Brief explanation of each python file:

1. src folder:
   a) **xval.py** specifies the class XvalMerge that helps collect all the important information during the training process
   b) **vi.py** specifies the log-likelihood of the observed data: log [p(x | theta) p(theta) / q (theta | x)]
   c) **utils.py** specifies some useful functions for getting the value of specific data/ file and directory management
   d) **solvers.py** specifies a function that solves an ODE numerically using python built-in control flow (for loop)
   e) **procdata.py** specifies useful classes/functions for data preprocessing
   f) **parameters.py** specifies useful classes to parametrize the model
   g) **encoder.py** specifies the conditional encoder
   h) **distribution.py** specifies all the possible distribution (Chaineddistribution class) that we might need and sampling method (Dotoperator class) for all the parameters we need (functions like: local, global and conditional, global, constant)
   i) **decoder.py** specifies the decoder part of the paper that can reconstruct the data from latent space

j) **<u>convenience.py</u>** defines all the upper level Classes that we will directly use in the run_xval.py
k) **<u>run_xval.py</u>** is the file where we can start one iteration of k-fold validation training process by running the main function in the run_xval.py. It specifies the pipeline of the whole training process
l) **<u>call_run_xval.py</u>** is the file where we can start the whole k-fold validation training process by running the main function in the call_run_xval.py. It executes the whole k-fold validation training process.
m) **<u>result</u>** is a subfolder of **src** folder, there are **.npy** files storing experiments datas and also files **events.out.tfevents** for tensorboard visualization. Here you should specify the path of the folder that contains these files for the tensorboard to visualize the training process.

2. models folder
   a) **<u>base_model.py</u>** defines the base model and we build a lot of different version upon that base model
   b) **<u>dr_blackbox.py specifies</u>** a lot of different version of models that inherit from the base model
3. data folder
   a) csv files that contain all the data needed for training the model
4. specs folder
   a) **<u>dr_blackbox_xval_hierarchical.yaml</u>** gives the specification for the training process and hierarchical blackbox model corresponding to the class ***DR_HierarchicalBlackbox*** in the dr_blackbox.py file