DC Neural ODE API

- Anshuman Mishra (shivance@github)

A. Modeling:

The goal is to provide a general Neural ODE API which can be used as an out-of-box model for modeling any physical system in general.

Dependency: Torchdyn for ode solvers

Features:

1. Using the API user should be able to fit the out-of-the-box ODE Model on time series out of the box.
    a. Solution :
        i. A TorchModel wrapper to fit the model with Deepchem's inbuilt or user made custom datasets say **NeuralODE**
2. User should be able to provide custom Neural Network (for ode) which represent vector field
    a. Solution :
        i. Allowing the user to pass the nn.Module through parameter in the **NeuralODEModel**, which in turn would pass the nn.Module to Torchdyn's **NeuralODE** and instantiate the desired neural vector field model.
3. There may be a use case where deepchem's ODE model cannot be fitted directly and would require users to implement their custom data generator and fit methods. Users should be able to do so with minimal changes.
4. Letting users choose their ODE Solver related specs like solver, adjoints, sensitivity method, seminorms, and integral_losses etc.
    a. Solution :
        i. Torchdyn's NeuralODE layer allows us to specify solver related specs. We provide these parameters in our top level API i.e. **NeuralODEModel**

B. Evaluation Metrics:

Users should be able to evaluate the model's performance.

Solution:

1. Deepchem's metrics API allows us to measure and state a model's performance.

What are some of the common problems encountered?

1. Searching for an architecture for neural networks in the neural ode class which performs high.

How different is Deepchem's ODE API from DeepXDE ?
DeepXDE provides a far bigger API to solve time dependent and independent PDEs, integro differential equations. It also has support to solve various kinds of differential equations. DeepXDE lets you define the differential equation and solves everything out-of-the-box !

Deepchem's ODE API apart from providing the feature to solve the defined differential equation let's users choose their own solvers, adjoint methods, sensitivity methods and even their own neural network to model the equation, by providing through parameters. Deepchem's Dataset class is more generic and lets users create their own type of dataset for their user case with minimal code.

Pseudo Code :

class NeuralODEModel(TorchModel):

__init__():
Params :
1. Parameters for TorchModel
2. solver
3. Order
4. Atol
5. Rtol
6. Sensitivity
7. Solver_adjoint
8. Atol_adjoint
9. Rtol_adjoint
10. Interpolator
11. Integral_loss
12. Semi_norm

override other torchmodel methods to create custom generator, and fit methods