GSoC’22 Proposal (First Draft)

Better Stack Layering

This is the large project (~350 hours) category. This project will introduce breaking changes in ANN API.

DAG Structure

- The structure will contain all layers of certain ANN.
- In this approach, we are aiming for multiple input, multiple output network should be generated easily.
- We will use existing layer structure mostly for forward and backpropagation.
- We will connect these layers through two types of pointers:
  - Next Pointer - It will point to next layer(s) of ANN. Used for DAG operations and backpropagation.
  - Prev Pointer - It will point to prev layer(s) connected to current layer. Can be used for detecting dangling layer (Tentative).
- For forward and backward propagation, output can be stored inside layer itself, and can be accesssed by any layer or model wrapper itself.
- We will introduce 2 new types of layers:
  - Input Layer - It will help in facilitating input, and identifying input point of ANN architechure. It takes input of size of single object as an array. This may also help verifying the network stability by size matching.
  - Output Layer - It will tell the output point of ANN architecture. May help in multi-objective learning. (Not researched deeply.)

DAG Class

This class will be used for training and evaluation of data on ANN architechure. It will take input nodes, output nodes, optimizer, etc. It contains following methods:

`Compile`

- It will be used to validate and generate metadata for ANN architecture.
- It will first run DFS on ANN architecture to detect different layers in current architecture. (It may also help in telling if a output node is unreachable, or a layer has some linking issue.)
- Then, it will run Topological sort with grouping such that it can help in training ANN. We have decided to use this algorithm so that with grouping, it will help in parallelizing the network in future. (It will also be used to detect cycles.)
- Then we will run a mock forward propagation to test dimension before training.

`Forward`

- This method will be used to run forward propagation in ANN architecture.
- It will use the fact that for any in layer in a given group i in topological sort, it will have atleast one dependency in previous group.
- So, we will do propagation group wise, i.e., grp_0 -> grp_1 -> ... -> grp_n.

```
Algorithm Forward:
Input : A DAG G with input X
for grp in topo_grp_ordered_set:
    for layer in grp: /* Can be parallelized as it doesn't depend on same grp layers */
        Forward(layer)
```

Correctness can be seen using induction on ordering of group set of topological sort.

`Backward`

- This method will be used to run backward propagation in ANN architecture.
- It will use the fact that for any in layer in a given group i in topological sort, it will have atleast one dependency in next group.
- So, we will do propagation reverse group wise, i.e., grp_n -> grp_{n-1} -> ... -> grp_1 -> grp_0.

```
Algorithm Backward:
Input : A DAG G with input Y
for grp in reverse(topo_grp_ordered_set):
    for layer in grp: /* Can be parallelized as it doesn't depend on same grp layers */
        Backward(layer)
```

Correctness can be seen using induction on reverse ordering of group set of topological sort.

`ShowModel`

This method will be used to display model description in meaningful way. Requires more thought about this.

`SaveModel`

This method will be used to save model’s weights.

`LoadModel`

This method will be used to load model’s weights.

Pre-trained models

This is the large project (~350 hours) category. It will introduce many architecture for ANN models. We will use shubham1206agra/**pretrained-models.pytorch** which is a fork from Cadene/**pretrained-models.pytorch**. I have ironed out some bugs from original repo and it contains many pre-trained model on pytorch. We will use kartikdutt18/**mlpack-PyTorch-Weight-Translator** to convert weigts from pytorch model. Since it is also BSD-3-Clause licensed, same as mlpack, it will be compatible with mlpack use license wise too. Main work would be to implement exact same model.

List of models can be seen here.