COMP 423/523:
Computer Vision for Autonomous Driving
Homework 2

Due: May, 8 2022

Introduction

In this assignment, you will first answer some questions based on the concepts and the algorithms that we learned in the class. Then, you will implement the missing parts of the VectorNet [2] architecture in the template code attached.

1. (**20 points**) Pen-and-paper part
2. (**30 points**) SubGraph implementation
3. (**30 points**) GlobalGraph implementation
4. (**20 points**) Loss computation

Important Note: This is an individual assignment, not a group work. You can discuss your solutions with each other but you are supposed to work on the code on your own.

0.1 Setup

To run the code we provide to you, you need to do a few things.

1. Clone the argoverse repository to your local/remote machine with the command:
   `git clone https://github.com/argoai/argoverse-api.git`
2. Download the file "hd_maps.tar.gz" from this link:
   https://s3.amazonaws.com/argoai-argoverse/hd_maps.tar.gz and extract it inside the argoverse-api directory.
3. Add the cloned repository to your python environment using the command:
   `pip install -e /path_to_root_directory_of_the_repo/`
   (replacing the `/path_to_root_directory_of_the_repo/` with the correct path)
4. Install pytorch and numpy if you don't have them installed already.

1 Pen-and-Paper Questions

1. (**2 points**) What kind of map representations are used in prediction? Compare them to each other by listing advantages and disadvantages of each.
2. (**2 points**) Explain the motivation behind the multi-scale lane convolution in the LaneGCN that we have seen in the lecture. Please read the paper and point to performance gains due to multi-scale specifically by referring to the corresponding experimental results (tables, figures, etc.) in the original paper.
3. (**2 points**) What is the difference between the SubGraph and GlobalGraph in VectorNet? Explain the role of each in the hierarchy intuitively.
4. (**2 points**) Explain the map completion loss of the VectorNet and point to the corresponding results in the paper to explain its effect on the performance.
5. (**2 points**) Compare VectorNet representation to LaneGCN that we have seen in the class. Please make sure to compare the two approaches experimentally as well by referring to the results in both papers in terms of various metrics as well as the the inference speed.
6. (**2 points**) What is the role of targets in anchor-based methods such as TNT? What is the danger of target-based solution of TNT, propose a fix. (**Hint:** Check the papers that cite the TNT on google scholar.)
7. (**4 points**) Explain the following concepts in the context of VAEs: prior, data likelihood, posterior, recognition model, reconstruction term, KL divergence.
8. (**2 points**) Explain the need for re-parametrization trick in the VAEs.
9. (**2 points**) Explain the role of KL term and the reconstruction term in the stochastic video generation.

2 VectorNet

The aim of this section is to complete the implementation of VectorNet [2]. The template code we shared with you in this github repository: `https://github.com/egeonat/cvad_hw2` contains five files, you need to modify only the two of them:

- **GNNs.py:** In this file, you will implement the GlobalGraph and SubGraph classes.
- **Vectornet.py:** You will initialize your GlobalGraph and SubGraph instances in this file, and also complete the loss computation. **run.py:** This is where the training code is implemented, you will be using this file to train & validate your implementation. You might need to take a look at this file to see which command line arguments are available.

- **preprocess_data.py:** We used this file to convert the Argoverse data to the VectorNet format. We provide it in case you want to take a look but you do not need to edit or use this file.

- **utils.py:** This file contains various utility functions used by the other files, feel free to take a look at them to understand the code, but you do not need to use them.

The training and validation data for training VectorNet is generatted by processing the Argoverse [1] dataset. We have already done this for you, and the files can be found on the HPC cluster under the paths:
`/userfiles/eozsuer16/argo/ex_list`
`/userfiles/eozsuer16/argo/eval.ex_list`
You will only need to provide these paths as commandline arguments to the run.py file.

The Data

We first explain the data format here. The entire dataset is inside these two pickled lists. Each item in these lists is a scene represented by a dictionary. However, due to large memory requirement of keeping all the scenes in memory, each dictionary is compressed and needs to be decompressed. The code that does this is provided for you in the Dataset class in utils.py.

The keys in the scene dictionary and their values are as follow:

- `file_name`: csv file of the given scene from raw data
- `start_time`: Start time of the scene
- `city_name`:City that the scene belongs to
- `cent_x`: x coordinate of the agent @2s
- `cent_y`: y coordinate of the agent @2s
- `angle`: Angle in radian of the agent @2s
- `two_seconds`: Timestamp value of 20th sample
- `agents`: List of past trajectories of tracks in the scene. Each item is a numpy array of shape `(existing_sample, 3)`. Row formation is `[x_coordinate, y_coordinate, timestamp]`. Note that 0th index corresponds to agent of interest.
- `matrix`: All vectors in the dataset in a stacked format. These vectors are minimum entities in the VectorNet. Numpy array of shape `(vectors, feature_dim)`.
- `polyline_spans`: Vector occupancies of polylines. `polyline_spans[i]` corresponds to lower and upper row indices as a slice in the matrix for the polyline i.
- `labels`: Future trajectory of agent of interest, which is the ground truth.

2.1 SubGraph

In this part, you will need to implement the SubGraph module of VectorNet[2]. This module treats every vector in a polyline as a node, and aggregates the information in each node through a GNN. In the end, features are pooled and a unified representation for the polyline is generated. You can refer to the section 3.2 of the paper for the implementation details of this module.

2.2 GlobalGraph

In this part you will need to implement the GlobalGraph module of the VectorNet[2]. This module treats every polyline in the scene as a node, and aggregates the information in each node through another GNN. In the end, features are pooled and a unified representation for the entire scene is generated. You can refer to the section 3.3 of the paper for the implementation details of this module.

2.3 Loss Calculation

Finally, you will need to implement the loss function required to train your model. You can refer to the paper for details on the loss function.

Note: Remember that even though the network generates multiple candidate predictions, only the best prediction is used for loss calculation.

Training

You can use the run.py file to train your model. You do not need to evaluate your model separately, as the training script periodically evaluates your model. You do not need to train your model until it converges while you develop it because it will take too long. You can check your results after training for a couple of epochs epochs. Discuss your results briefly in your report.

You can expect a miss rate (MR) of around 0.62 after 2 epochs of training.

Submission

Your report should be in the pdf format and your code should be hosted as a GitHub repository. Include the link to your GitHub project at the start of your report. You will then only submit your pdf file to blackboard.

Bonus (10 points): Map Completion

For the bonus part, you will be implementing the map completion auxiliary loss we skipped previously. Here are the necessary steps to implement this:

- Create a coordinate-based ordering of polylines, and append this to the polyline features.
- Write a random mask function to randomly mask out a polyline from the scene global graph.
- After the masked out polyline features have been updated through the global graph, pass them through an MLP.
- Calculate the auxiliary loss using the Huber loss function.

Important: Please make sure your GitHub repositories are private and add the TA as a collaborator (username is egeonat).

References

[1] Ming-Fang Chang et al. "Argoverse: 3D Tracking and Forecasting with Rich Maps". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2019.

[2] Jiyang Gao et al. "VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2020.