To Reviewer 1: Thanks for your review. (I) Fair comparisons: the datasets in our evaluation come from ASTGCN and STSGCN. Our results can be cross-checked to examine the performance (e.g., compare ref [11]). The code and datasets are also provided for reproducing our results. For other baselines (e.g., DCRNN, STGCN), we fine-tuned them to choose the best settings and hype-parameters to ensure our comparison is fair. We believe our results are solid and the superiority can be explained from two views: first, the predefined graph, which is provided by ASTGCN and STSGCN, may be sub-optimal compared to our adaptively learned graph; second, the NAPL module can do unique fitting for each node effectively. (II) Parameter size: we compared the number of parameters in Table 3 (page 8), which shows that AGCRN has moderate parameters when compared with baselines (e.g., ASTGCN). Besides, if setting the embedding dimension to a smaller value, AGCRN will contain much fewer parameters but still achieve good performance (as shown in Figure 4). (III) E_g: there is no need to force E_g be orthogonal since an identity matrix is added to the GCN. (IV) Learned graph: the values in the learned adaptive graph vary from 0.0014 to 0.0396 (for PeMSD4), which demonstrates that DAGG can reveal relative importance among different series. We will visualize the learned adaptive graph with heatmap and plot some highly-correlated series in the supplementary. We can achieve sparse graph and avoid negative links by setting small values to zero. However, we didn't see it brings obvious improvements in our experiments. The learned graph in DAGG can perform well and we will keep exploring more advanced graph generation methods.

To Reviewer 2: Thanks for your careful review. (I) Experiments: In Table 1, the "*" sign should be presented to the results of STSGCN. As you have listed, DCRNN, STGCN, and ASTGCN used different datasets for evaluation. We followed the most recent works (e.g., ASTGCN and STSGCN) and conducted experiments on the PeMSD4 and PeMSD8 datasets. Our results can be cross-checked and compared with the results presented in ASTGCN and STSGCN. (II) Metr-LA: the Metr-LA dataset is not the first choice for our evaluation because prior works in Metr-LA (e.g., DCRNN) use early fusion (e.g., concatenation) to integrate the extraneous data (e.g., time of day), which is not the focus of our work and does not help the graph generation process. However, we still conduct experiments on Metr-LA following your suggestion and compare our results with the reported results in DCRNN (will add to the supplementary). Experiments show that AGCRN achieves comparable performance to DCRNN and consistently outperforms GCRNN (i.e., DCRNN with undirected graph). The long-term prediction performance of AGCRN is especially superior (for 1 hour ahead prediction, MAE:3.57, RMSE:7.38, MAPE:10.09%). Our further analysis show AGCRN can gain more advantages over DCRNN under a fair comparison (i.e., with the extraneous data removed for both models), even though AGCRN does not rely on encoder-decoder, teacher forcing, or learning rate decay strategies while DCRNN does. (III) Our preliminary experiments show that pair-wise distance or similarity-based graph can work together with DAGG under a multi-graph framework and provide extra information. Also, we agree that generating outputs in a sequential manner (e.g., under the encoder-decoder framework) may improve the results despite sacrificing efficiency. It is convincing that AGCRN has potentials to enable more advanced performance in time series analysis. (IV) Indeed, learning in large graphs for evolving systems is important yet not well studied in literature. Graph partition and sub-graph training may probably work with some adaption. We will highlight this direction as a future work.

To Reviewer 3: Thanks for your comments. (I) Loss function: unfortunately, there is no standard about which loss function is better. DCRNN uses L1 loss; STSGCN uses Huber Loss; STGCN uses L2 loss. We chose L1 loss because it achieved better results in our experiments than L2 loss. We would like to highlight that we have carefully chosen the best settings (e.g., loss function) for the baselines to make sure our comparison is fair unless stated otherwise. (II) Prediction Length: first, each step is 5 mins in our experiments (see line 196 of page 5). Thus, we predict the future 60 mins data with the historical 60 mins data, which is the most common setting used for traffic forecasting. Second, our problem is formulated to predict several future steps data with multiple historical steps data (Section 3.1). Thus, the exact time lengths of the historical time period and future period are dependent on the number of time steps and the length of one step, which could vary by applications. (III) Multi-step prediction: Yes, the results in Section 4.4 come directly from the results of multi-step prediction. As formulated in Section 3.1 and mentioned in Section 3.5, our method can generate multi-step predictions directly. With 5 mins as the slot length in our experiments, the short-term (5 mins) and long-term (60 mins) predictions refer to the predictions made at the first step and the 12^{th} step, respectively.

To Reviewer 4: Thanks for your comments. (I) we omitted some works about GCN for classification tasks due to the space limitation and different problem settings. We will add a subsection under Section 2 to discuss those omitted studies. Specifically, our work deal with dynamically evolving streams from nodes in the whole graph but GCN-based classification models deal with static features/attributes from nodes within a neighborhood. Our work may share some high-level similarities with these studies but has totally different designs for different purposes. For NAPL, our work aim to learn unique parameters for each node, while GAT aims at learning different importance scores for neighbors. The weights in GAT are still shared among different sub-graphs. The matrix factorization in NAPL may be similar with clustering in graph pooling at a high level, but they work in different ways. For DAGG, the learned node embedding in our work can be regarded as high-level abstraction of the whole data stream and is used to generate the graph. But in GCN-based classification models, the learned node embedding is the transformation of nodes feature with the help of existing graph. (II) we apologise for the typos. Metr-La should be PeMSD8 and STSGCN should be ref [11]. We have revised them and conducted a thorough proofreading to eliminate possible typos and grammar mistakes.