Lite-QCNet: Technical Report for 2024 Argoverse Motion Forecasting Challenge

Zikang Zhou[1] Zihao Wen[1] Jianping Wang[1] Yung-Hui Li[2] Yu-Kai Huang[3]
[1]City University of Hong Kong [2]Hon Hai Research Institute [3]Carnegie Mellon University

Abstract

This technical report introduces Lite-QCNet, a lightweight joint multi-agent motion forecasting model. This model improves the efficiency of QCNet/QCNeXt in two ways. First, it utilizes a patching module for temporal abstraction, which reduces the number of tokens in the time dimension and improves the efficiency of factorized spatial-temporal attention. Second, we use k-NN local attention for spatial interaction modeling, which enables more stable inference latency and memory usage than radius-based local attention. The resulting model consumes much fewer training resources than QCNet/QCNeXt while achieving similar prediction performance on the Argoverse 2 Multi-Agent Motion Forecasting Benchmark.

1. Introduction

Efficient motion forecasting is critical for the safety of autonomous driving. While the prediction accuracy of state-of-the-art motion forecasting models has been continuously improved by introducing advanced architecture, *e.g.*, Transformers [3], the efficiency issue is becoming more and more prominent. In this technical report, we revisit the state-of-the-art spatial-temporal Transformer models for motion forecasting, *i.e.*, the QCNet family [6, 7], and introduce two improvements to lower the training/inference latency and the memory usage. First, inspired by [5], we utilize a patching module to reduce the temporal tokens, making the spatial-temporal Transformers more efficient. Second, we replace radius-based spatial attention with k-NN spatial attention, which can stabilize inference latency and memory usage. By incorporating these modifications, our proposed Lite-QCNet performs similarly to QCNet/QCNeXt on the Argoverse 2 Motion Forecasting Dataset [4] with lower training and inference costs.

2. Methodology

This section describes our modifications to the QCNet family [6, 7]. Previously, QCNet and QCNeXt utilized the relative spacetime representation to incorporate roto-translation invariance in space and translation invariance in time. As a result, these models can achieve lower inference latency by leveraging the key-value cache for Transformers. However, implementing feature caching requires substantial engineering efforts. On the other hand, the radius-based local attention modules in QCNet/QCNeXt may suffer from high costs in extreme scenarios, such as those involving heavy traffic. This work aims to arrive at an efficient solution even without feature caching (although feature caching can still be supported as long as we respect the invariance in space and time), especially when the historical observation window is large. Our modifications to the baseline models are illustrated as follows.

2.1. Modification 1: Agent Patching

For a Transformer architecture that factorizes space and time, the temporal self-attention has the complexity of $\mathcal{O}(AT^2)$, the agent-map cross-attention has the complexity of $\mathcal{O}(ATM)$, and the agent-agent self-attention has the complexity of $\mathcal{O}(A^2T)$, with A, T, and M denoting the numbers of agents, historical time steps, and map instances, respectively. Without the key-value cache, all these attention modules would be rather expensive.

Inspired by BehaviorGPT [5], we introduce an attention-based patching module for temporal abstraction, leading to fewer temporal tokens of agents. Specifically, we define a trajectory patch of an agent as

$$P_i^{\tau} = S_i^{(\tau-1)\times\ell+1:\tau\times\ell}\,,\; i \in \{1,\ldots,A\}\,,\\ \tau \in \{1,\ldots,N_{\text{patch}}\}\,, \tag{1}$$

where ℓ is the number of time steps covered by a trajectory patch, $N_{\text{patch}} = T/\ell$, S_i^t denotes the i-th agent's state at time step t, and P_i^{τ} represents the τ-th patch of the i-th agent. On top of this, we utilize an attention module to obtain patch-level embeddings:

$$\hat{P}_i^{\tau} = \text{MHSA}(Q = \hat{S}_i^{\tau\times\ell}, K = V = \{[\hat{S}_i^t, \mathcal{R}_i^{t\to\tau\times\ell}]\},\\ t \in \{(\tau-1)\times\ell+1,\ldots,\tau\times\ell-1\})\,. \tag{2}$$

Here, $\hat{S}_i^t$ is the embedding of S_i^t, $\hat{P}_i^{\tau}$ is the patch embedding of the i-th agent at the τ-th patch, $\text{MHSA}(\cdot)$ is multi-head self-attention, $[:,:]$ denotes concatenation, and $\mathcal{R}_i^{t\to\tau\times\ell}$ is the positional embedding of S_i^t relative to $S_i^{\tau\times\ell}$.

Table 1. Single-Model Prediction Results on the Argoverse 2 Validation Set

Method	b-minSFDE$_6$ $\downarrow$	minSADE$_6$ $\downarrow$	minSFDE$_6$ $\downarrow$	actorMR$_6$ $\downarrow$	actorCR$_6$ $\downarrow$
QCNeXt	1.78	0.52	1.12	0.14	0.80%
Lite-QCNet	1.78	0.53	1.12	0.14	0.77%

Table 2. Quantitative Results on the Argoverse 2 Multi-Agent Motion Forecasting Benchmark

Method	b-minSFDE$_6$ $\downarrow$	minSADE$_6$ $\downarrow$	minSFDE$_6$ $\downarrow$	actorMR$_6$ $\downarrow$	actorCR$_6$ $\downarrow$	minSADE$_1$ $\downarrow$	minSFDE$_1$ $\downarrow$
HeteroGCN	2.12	0.69	1.46	0.19	0.01	1.23	3.05
FJMP	2.59	0.81	1.89	0.23	0.01	1.52	4.00
Forecast-MAE	2.24	0.69	1.55	0.19	0.01	1.30	3.33
FFINet	2.44	0.77	1.77	0.24	0.02	1.24	3.18
QCNeXt×8	**1.6536**	0.5003	1.0232	0.1310	0.0088	0.9372	2.2866
QCNeXt×4+Lite-QCNet×4	1.6664	**0.4978**	**1.0154**	**0.1282**	0.0088	**0.9287**	**2.2656**

The complexity of this patching operation is $\mathcal{O}(AT)$. Using this patching operation, we assimilate the embeddings of $S_i^{(\tau-1)\times\ell+1:\tau\times\ell-1}$ into that of $S_i^{\tau\times\ell}$ and acquire the patch embedding $\hat{P}_i^{\tau}$. Subsequently, the temporal self-attention, the agent-map cross-attention, and the agent-agent self-attention all operate on the patch-level embeddings of agents. As a result, their complexities will become $\mathcal{O}(AT^2/\ell^2)$, $\mathcal{O}(ATM/\ell)$, and $\mathcal{O}(A^2T/\ell)$, respectively, where ℓ is the number of time steps in a patch. We set ℓ as 5 in our implementation without hyperparameter tuning, but using a larger patch size may further lower the complexity without affecting the performance.

2.2. Modification 2: k-NN Attention

Spatial interaction modeling is essential for understanding traffic scenarios. For this reason, many motion forecasting models incorporate modules like self-attention among map elements, self-attention among agents, and cross-attention between agents and map elements. Since the number of tokens involved in the attention operation may be large, we can utilize local attention to reduce the complexity. QCNet selects a query element's neighboring key/value elements according to some distance thresholds, which results in a varied number of neighbors and unstable computational/memory costs. To mitigate this problem, we use k-NN attention modules instead, making the computational and memory costs more controllable. For instance, the complexity of agent-map cross-attention and agent-agent self-attention in the scene encoder will be further reduced to $\mathcal{O}(ATk/\ell)$ by the k-NN neighbor selection. On the other hand, the query-based decoder will have the complexity of $\mathcal{O}(ANk)$ for map attention and agent attention, where N is the number of modes. We set the number of neighbors for all spatial attention modules in the encoder as 32. On the other hand, the number of neighbors is set as 100 in the decoder. Again, these hyperparameters are chosen according to our intuition without trial and error.

3. Experiments

3.1. Implementation Details

We set the hidden size as 128 and obtain models with 8.3M parameters. Most implementation details in model architecture follow QCNeXt [6,7], except we have a patching module at the very beginning of the encoder and employ k-NN attention in both the encoder and the decoder. We train models on the training split of the Argoverse 2 Motion Forecasting Dataset [4] using the AdamW optimizer [2]. The training process lasts 50 epochs, with a batch size of 32, a weight decay rate of 0.1, a dropout rate of 0.1, and an initial learning rate of 5×10^{-4} decayed to 0 by the cosine annealing scheduler [1].

3.2. Ensembling

Our test set results are obtained via the ensemble of four Lite-QCNet models and four QCNeXt models [7]. These models are trained using different random seeds. Given the prediction results produced by these models, we use k-means ensembling for trajectory aggregation, as illustrated in [7].

3.3. Quantitative Results

The single-model prediction results on the validation split of the Argoverse 2 Motion Forecasting Dataset [4] are shown in Tab. 1. From the results, we can conclude that Lite-QCNet performs similarly to QCNeXt, with a slightly lower collision rate and a slightly higher average displacement error.

The test set results of the Argoverse Multi-Agent Motion Forecasting Benchmark are listed in Tab. 2. On the one hand, both QCNeXt and the combination of QCNeXt and Lite-QCNet outperform other approaches significantly. On the other hand, the combination of QCNeXt and Lite-QCNet outperforms pure QCNeXt on all metrics except the Brier Score.

In summary, although Lite-QCNet consumes fewer computing resources than QCNeXt, it performs competitively in multi-agent motion forecasting, demonstrating the effectiveness of our improvements.

4. Conclusion

In this technical report, we rethink the architecture design of QCNet/QCNeXt and derive a lightweight variant named Lite-QCNet. Benefiting from the patching module and the k-NN attention, we significantly improve the efficiency of the baseline model while achieving state-of-the-art performance on the Argoverse 2 Multi-Agent Motion Forecasting Benchmark.

References

[1] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 2

[2] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 2

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 1

[4] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks)*, 2021. 1, 2

[5] Zikang Zhou, Haibo Hu, Xinhong Chen, Jianping Wang, Nan Guan, Kui Wu, Yung-Hui Li, Yu-Kai Huang, and Chun Jason Xue. Behaviorgpt: Smart agent simulation for autonomous driving with next-patch prediction. *arXiv preprint arXiv:2405.17372*, 2024. 1

[6] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-centric trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2

[7] Zikang Zhou, Zihao Wen, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Qcnext: A next-generation framework for joint multi-agent trajectory prediction. *arXiv preprint arXiv:2306.10508*, 2023. 1, 2