Multi-level Community-awareness Graph Neural Networks for Neural Machine Translation

Binh Nguyen[1,2], **Long Nguyen**[1,2]*, **Dien Dinh**[1,2]
[1]Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam
[2]Vietnam National University, Ho Chi Minh City, Vietnam
nqbinh17@apcs.fitus.edu.vn, {nhblong,ddien}@fit.hcmus.edu.vn

Abstract

Neural Machine Translation (NMT) aims to translate the source- to the target-language while preserving the original meaning. Linguistic information such as morphology, syntactic, and semantics shall be grasped in token embeddings to produce a high-quality translation. Recent works have leveraged the powerful Graph Neural Networks (GNNs) to encode such language knowledge into token embeddings. Specifically, they use a trained parser to construct semantic graphs given sentences and then apply GNNs. However, most semantic graphs are tree-shaped and too sparse for GNNs which cause the over-smoothing problem. To alleviate this problem, we propose a novel Multi-level Community-awareness Graph Neural Network (MC-GNN) layer to jointly model local and global relationships between words and their linguistic roles in multiple communities. Intuitively, the MC-GNN layer substitutes a self-attention layer at the encoder side of a transformer-based machine translation model. Extensive experiments on four language-pair datasets with common evaluation metrics show the remarkable improvements of our method while reducing the time complexity in very long sentences.

1 Introduction

Self-attention mechanisms, introduced by Transformers (Vaswani et al., 2017), have been ubiquitous in Neural Machine Translation (NMT). It encodes the relative information of a position based on other positions in the same sequence. Recent works have seen the rising trend of improving the self-attention module such as (Choromanski et al., 2021; Katharopoulos et al., 2020).

One of the serious self-attention weaknesses is its inefficiency at computing long sentences due to quadratic complexity. Following this problem, Performer (Choromanski et al., 2021) estimates a full-rank-attention matrix with Fast Attention Via positive Orthogonal Random features approach (FAVOR+), that trade-offs the time complexity from $O(T^2d)$ to $O(Td^2logd)$. However, as they scale the model larger such as GPT-3 (Brown et al., 2020), the quadratic dimensional space d exponentially scales much faster than that of the sequence length T. In addition, Linear Transformer (Katharopoulos et al., 2020) reported a comparable performance with the Transformer, while speeding up on autoregressive inference time with a linear dot-product of kernel feature maps. Several studies replace self-attention with an attention-free module, which significantly speeds up the training time. Attention Free Transformer (ATF) (Zhai et al., 2021) eliminates the dot-product attention with weighted element-wise multiplication, which linearizes the time complexity to $O(Td)$. FNet (Lee-Thorp et al., 2021) is a pre-trained model with Discrete Fourier Transform that speeds up 80% training time on GPU but still achieves 92-97% of the accuracy of BERT (Devlin et al., 2018). These works make a trade-off between the accuracy and the performance to reduce the computational cost.

Another line of work constructs a content-based sparse graph to select keys that have high similarity meanings to queries. Routing Transformer (Roy et al., 2020) applies k-means clustering to cluster both queries and keys. Meanwhile, Reformer (Kitaev et al., 2020) uses locality-sensitive hashing (LSH) to select key-value pairs for each query.

Unlike previous work, in this paper, we follow the intuition that self-attention is assumption-free on the structural input, and thus it is hard to induce task-based generalization on small-scale data. As a result, we propose semantic-based sparse attention that utilizes the semantic graph construction of the Universal Conceptual Cognitive Annotation (UCCA) (Abend and Rappoport, 2013) parser, which is implemented based on Message Passing Neural Networks (MPNN). The vital benefit of

*Corresponding author

Proceedings of the 29th International Conference on Computational Linguistics, pages 5021–5028
October 12–17, 2022.

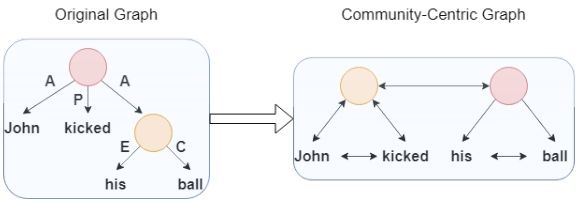


Figure 1: A UCCA graph (left) for sentence "John kicked his ball" (Abend and Rappoport, 2013) and its community-centric graph (right). The colored circles are intermediate nodes.

UCCA is that it is not relatively defined to a specific language, or in another word, it provides a universal structure across languages. We hypothesize that the translation task will benefit from this characteristic. According to (Abend and Rappoport, 2013), because UCCA is a connected acyclic graph (or tree) whose in-degree node usually is one or two, the attention-based computation will smooth the token embeddings to a high similarity score. This problem is referred to as the over-smoothing problem, leading to the bad convergence when the MPNN model is trained deeper.

To alleviate the issue, we introduce a set of transformation rules to convert the tree-based UCCA into a Community-Centric Graph (CCG). As illustrated in Figure 1, the CCG (on the right) presents two different communities that somehow discover local relationships between words and word roles in a community. Moreover, we can theoretically prove that the CCG structure is stronger than the tree-based UCCA (see Section 3.2). To capture the global relationships across communities, we propose Multi-level Explorations (ME) to probe multi-level structures in a graph. Henceforth, we combine CCG and ME into Multi-level Community-awareness Graph Neural Networks (MC-GNN). This MC-GNN layer will substitute a self-attention layer in the Transformer encoder.

MC-GNN is theoretically faster than the Transformer, thanks to the sparse attention. Table 1 show the time complexities for the Trasformer variants. Experimental results showed that MC-GNN outperforms the Transformer-based MT (Vaswani et al., 2017) in three over four NMT benchmarks across four evaluation metrics (Section 4.2). Moreover, we observe that Performer, Linear Transformer, ATF, and FNet are underperformed in translation tasks. We also report that the time complexity is equal to at most 60% and 10% of the self-attention

Table 1: Complexity comparision. Here T, d, and k denote the sequence length, hidden embedding, and the average node degree, respectively. $k << T$ for the sparse attention.

Model	Time
Transformer	$O(T^2d)$
FNET	$O(1)$
Performer	$O(Td^2logd)$
AFT	$O(T^2d)$
Linear Attention	$O(Td^2)$
MC-GNN	$O(Tkd)$

in short and long sentences, respectively (Section 4.3). The contributions of this work are as follows:

- We propose a Community-Centric Graph to provide community awareness and alleviate the over-smoothing problem in GNN.
- We propose Multi-level Explorations (ME) to probe multi-level structures in a graph.
- We prove the robustness and effectiveness of MC-GNN by extensive analyses.

2 Background and Related Work

2.1 Graph-based Approach in NMT

The explicit incorporation of linguistic information into traditional statistical machine translation has yielded many positive results (Bazrafshan and Gildea, 2013), and therefore it is intuitive to incorporate additional knowledge into NMT. Yet, governing restrictive constraints on the interaction between external information and the translation task will hamper MT performance. Bastings et al. (2020) introduced a graph-based encoder to effectively blend the syntactic structure into NMT. Meanwhile, Marcheggiani et al. (2018) embedded semantic bias into word representations by GNN, and thus provided semantic awareness to NMT. Xu et al. (2021) incorporated contextual awareness to document-level NMT by GNN, where a document is transformed into a graph that links relevant contexts (i.e., named entity) regardless of their distances. These methods, however, require a larger computational cost than the Transformer in the same settings.

2.2 Graph Attention Networks

Graph Attention Networks (Veličković et al., 2017) compute the representation of each node by the

weighted sum over its neighbors, following the graph-structured data. They introduce a set of node features as $h = \{h_1, h_2, ..., h_N\}$, $h_i \in R^D$ where N is the number of nodes, and D is the dimensional embeddings. To learn the higher-order information from the graph, they compute attention coefficients from two adjacent nodes as follows:

$$e_{ij} = LeakyReLU(a^T[Wh_i||Wh_j]) \quad (1)$$

e_{ij} is considered as the significance of node j to node i. Moreover, $a \in R^{2D}$ and $W \in R^D$ are trainable parameters, the *LeakyReLU* nonlinearity uses the negative slope $\alpha = 0.2$. They normalize the attention coefficients by using the softmax function as follows:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k \in N(i)} exp(e_{ik})} \quad (2)$$

where $k \in N(i)$ means for every neighbor of node i. Finally, the aggregated information of node i will be updated as follows:

$$\hat{h}_i = \sum_{k \in N(i)} \alpha_{ik} W h_k \quad (3)$$

The multi-head attention is employed to improve inductive learning, which is similar to Transformer (Vaswani et al., 2017).

2.3 Improvement of self-attention

Self-attention is the heart of many NLP-related deep learning tasks (Guo et al., 2019; Zhang et al., 2021; Peng et al., 2021; Lee-Thorp et al., 2021). Three profound elements matter in self-attention are: (1) computation complexity, (2) memory footprint, and (3) performance. Fundamentally, no research outperforms the Transformer in all three criteria in NMT because the trade-off between them always exists. For example, FNet (Lee-Thorp et al., 2021) replaces self-attention with an unparameterized Fourier Transform that reduces the training time by seven times on GPU and achieves 97% of the BERT-Base accuracy on GLUE. However, we find that FNet significantly underperforms against the Transformer in MT tasks. Random Feature Attention (Peng et al., 2021) linearizes the softmax function with random feature methods based on the Gaussian kernel (Rahimi and Recht, 2007). Especially they speed up the decoding time by two times and slightly improve the BLEU score by 0.1 BLEU in WMT14 En-De. Guo et al. (2019) substituted the fully-connected structure with a star-shaped topology, and thus time complexity is reduced from quadratic to linear. We, instead, replace the self-attention layer with MC-GNN and can gain better performance while reducing the complexity.

2.4 Universal Conceptual Cognitive Annotation

UCCA is a bi-lexical dependency graph whose goal is to abstract semantics away from syntactic interpretations in a typological and cross-linguistic fashion (Abend and Rappoport, 2013). To that aim, a multi-layer formalism is defined in UCCA, in which each layer specifies the relations it encodes. For example, the left graph of Figure 1 includes a single process scene (P), which describes an action or a movement that evolves in time. The process "kicked" contains two participants (A), "John" and "his ball". The participant "his ball" is further annotated with a center (C) and an elaborator (E).

3 Method

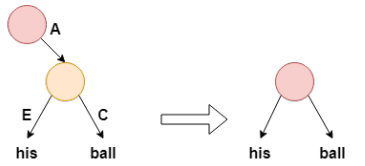
In this section, we firstly provide notations and theory of Community-awareness Graph (CCG). We then provide the proof that CCG is stronger than tree graph. We next present details of MC-GNN, and finally explain how MC-GNN is incorporated into a NMT model.

3.1 Graph Notations & Community-Centric Graph

Notation. Let G be a directed graph with N nodes and M edges. A non-symmetric adjacency matrix is defined as $A \in \{0,1\}^{NxN}$ and the embeddings of nodes are $X \in R^{NxD}$ where D is the length of embeddings. A node i points to a node j if and only if $A_{ij} = 1$. We denote $\hat{A} = A + I_N$ with I_N is an identity matrix, and $\hat{D} = diag(\hat{A}\mathbf{1})$ a degree matrix where $diag(\cdot)$ creates a diagonal matrix and $\mathbf{1}$ is the all ones vector. A normalized Laplacian matrix is computed by $\hat{L} = \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$.

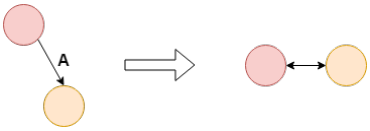
Community-Centric Graph. Let S: $A_U \rightarrow A_C$ be a set of transformation rules to convert a UCCA graph $G_U = (A_U, X)$ to a community-centric graph $G_C = (A_C, X)$. We propose three following rules that only apply to the UCCA graph.

- Same direction: $(A_C)_{ij} = 1$ if and only if (iff) $(A_U)_{ik} = (A_U)_{kj} = 1$ (Figure 2a).
- Opposite direction: $(A_C)_{ij} = (A_C)_{ji} = 1$ iff $(A_U)_{ui} = (A_U)_{uj} = 1$ (Figure 2b).



(a) Same direction

(b) Opposite direction



(c) Community link

Figure 2: Transformation rules

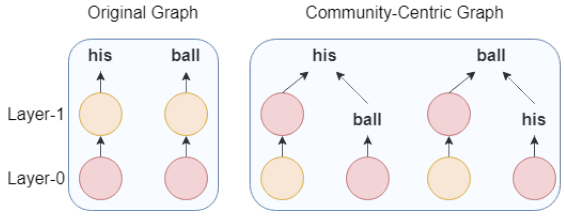


Figure 3: The left graph is indistinguishable after 1 layer, while the community centric graph (right) can be distinguished.

- Community link: $(A_C)_{ij} = (A_C)_{ji} = 1$ if $(A_U)_{ij}$ or $(A_U)_{ji} = 1$ and two nodes i, j are the intermediate nodes (Figure 2c).

In fact, the community-centric graph is inspired by a line graph (Gross and Yellen, 2005) that has been recently applied in (Zhao et al., 2020; Cao et al., 2021).

3.2 Community-Centric Graph is stronger than Tree Graph

Given an adjacency matrix A and node embeddings X, we define an attention-based GNN to compute graph-level embeddings as follows:

$$\hat{X} = GNN(A, X) \tag{4}$$

In the left-hand side of Figure 3, since the node *his* and node *ball* have the same set of neighbors, the weighted sum GNN of this set produces the equivalent embedding. As a result, the embeddings are undistinguished or over-smoothed after t layer GNNs. However, we have two different sets of neighbors in the community-centric graph, and thus the final embeddings are different.

3.3 Multi-level Explorations of Graph Structures

Motivated by rich closed-path sub-structures in the community-centric graph G_C, we propose a novel method that creates an adjacency matrix A_M to explore a multi-level structure in G_C. The profound idea is to utilize the adjacency matrix A_C as follows:

$$A_M^{(L)} = A_C^T \odot A_C^{(L)} \tag{5}$$

where L is the power of a matrix, $\odot$ is the element-wise multiplication. Intuitively, Equation 5 indicates that $(A_M^{(L)})_{ij} = 1$ if and only if there exists a path length L from node i to node j in $A_C^{(L)}$ and exists a path from node j to node i. To explore multi-level structures, we add matrix $A_M^{(t)}$ at each t^{th} encoder layer; if the standard Transformer has six layers then it probes six different levels.

Intuitively, the community-centric graph reveals the role of words in a local community, while the multi-level structure bright-to-lights the hidden role of words in global.

3.4 Multi-level Community-awareness GNN (MC-GNN)

Motivated by the multi-head self-attention that has been widely used in various NLP tasks, we introduce multi-head attention-based message propagation. Given a graph $G = (X^{(t)}, A)$ at a time step t, we update each node features $x_i^{(t)}$ as follows:

$$\begin{aligned} x_i^{(t+1)} &= GNN(x_i^{(t)}, A) \\ GNN(x_i^{(t)}, A) &= \sigma(W_1 x_i^{(t)} + F(x_i^{(t)}, A)) \end{aligned} \tag{6}$$

where σ is a PReLU activation function, and W_1 is trainable parameters. F is a scaled multi-head attention function defined in Equation 7.

$$\begin{aligned} F(x_i^{(t)}, A) &= \sum_{A_{i,j}=1} \alpha_{i,j} W_2 x_j^{(t)} \\ \alpha_{i,j} &= softmax(\frac{(W_2 x_i^{(t)})^T (W_2 x_j^{(t)})}{\sqrt{d}}) \end{aligned} \tag{7}$$

where W_2 is a matrix with trainable parameters, d is the dimensional embedding, and a multi-head annotation is abandoned for sake of simplicity. To apply MC-GNN, we first combine the multi-level graph $A^{(t)}$ and community-centric graph A_C, and then replace it in Equation 6 as follows:

$$\begin{aligned} A_{MC}^{(t-1)} &= A_M^{(t-1)} + A_C \\ x_i^{(t+1)} &= GNN(x_i^{(t)}, \hat{L}_{MC}^{(t-1)}) \end{aligned} \tag{8}$$

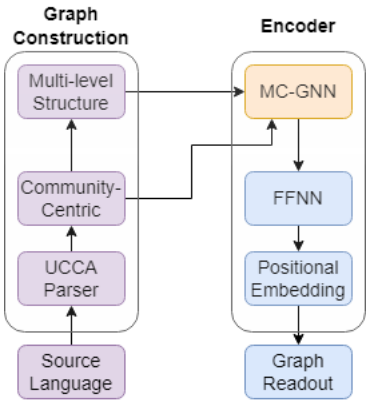


Figure 4: The Transformer-based NMT with the MC-GNN layer. Graph Construction is pre-processed and saved to files to speed up. Positional Embedding is added to token embeddings at the end of a layer yielding the better performance than added outside the Transformer block.

where $\hat{L}_{MC}^{(t-1)}$ is the Laplacian graph introduced in Section 3.1. Consequently, UCCA- and Community Centric- GNN replace the UCCA A_U and community centric A_C graphs in Equation 6. The computational complexity of the Transformer and the above methods will be examined in Section 4.3.

3.5 Model Architecture

Figure 4 illustrates the complete process and the model architecture when we incorporate MC-GNN into a NMT system. A source sentence is firstly passed through a graph construction process to convert the sentence to a graph structure. In this process, we can consider three different options: (1) to use the original UCCA, (2) to produce CCG using the transformation rules in Section 3.1, and (3) to aggregate multi-level structures as explained in Section 3.3. Next, based on the output from the graph construction process, graph information is produced by the MC-GNN layer and fed into Feed-Forward Neural Networks (FFNN). The positional information of token embeddings is mixed up with their neighbors during the message propagation, so positional embedding is added after the FFNN. The graph readout is necessary for removing irrelevant nodes and keeping token orders. Lastly, the decoder side is the same as the Transformer.

4 Experiments

We provide hyper-parameter settings, datasets, parser settings, and our code at https://github.com/nqbinh17/mc-gnn.

4.1 Settings

We experiment on four IWSLT language-pair datasets: English-French (En-Fr), English-Vietnam (En-Vi), English-Czech (En-Cz), and English-German (En-De). Moreover, we use a standard hyper-parameter for every model. The number of the encoder, and decoder layers is set to 6, the model embedding is 512, the intermediate size is 2048, the number of multi-head is 8, and the dropout is 0.3. For En-Cz, however, the numbers of the encoder and multi-head are set to 5 and 2, respectively.

4.2 Ablation Study

In this study, the performances of UCCA-, Community-Centric-, and MC-GNN are compared against the Transformer on the En-Fr dataset. Finally, we carry out the evaluations on the other three datasets to reveal the robustness of the MC-GNN.

- **Transformers** is the vanilla Transformer with the self-attention mechanism.
- **UCCA-GNN** is the Transformer-based model but replace self-attention with UCCA-GNN.
- **CC-GNN** and **MC-GNN** use the same model as UCCA-GNN but different graph constructions.

We first validate the argument in Section 3.2 on the tree-based (UCCA-GNN) graph and the community-centric (CC-GNN, MC-GNN) graph. Table 2 shows that CC- and MC- GNN improve UCCA-GNN by -0.71 and -0.76 BLEU in the 2014 testset. The outcome is impressive, because there is no change in the GNN architecture but with a simple graph pre-process.

On four language-pair datasets, we observe the consistent performances of MC-GNN as compared to the Transformer across four evaluation matrices. The improvements are substantial on En-Fr, En-Vi, and En-Cz, while the number of parameters are slightly lower. Nevertheless, MC-GNN consistently under-performed in the En-De dataset.

At first, we assumed that UCCA will be benefited from similar language pairs such as English, Vietnamese, and French, yet this argument fails in English-German that share similarities in grammar. Overall, MC-GNN is competitive to the Transformer in terms of performance and model size.

Table 2: The ablation study is carried out to show the effectiveness of the proposed method. The bold text is to highlight the least parameters or the highest performance of the model in a dataset.

Model	Dataset	Para. (M)	Test	GLEU	NIST	METEOR	BLEU
Transformer	IWSLT En-Fr	**128.0**	tst2014	37.77	7.51	56.75	35.76
			tst2015	38.32	7.48	56.64	36.16
UCCA-GNN	IWSLT En-Fr	124.9	tst2014	37.47	7.58	57.12	35.74
			tst2015	37.42	7.40	56.06	35.20
CC-GNN	IWSLT En-Fr	124.9	tst2014	38.17	7.65	57.5	36.45
			tst2015	38.47	7.53	56.92	36.23
MC-GNN	IWSLT En-Fr	124.9	tst2014	**38.22**	**7.65**	**57.50**	**36.50**
			tst2015	**38.51**	**7.55**	**57.10**	**36.35**
Transformer	IWSLT En-Vi	**82.4**	tst2013	31.74	6.91	N/A	28.23
			tst2015	29.05	6.39	N/A	26.31
MC-GNN	IWSLT En-Vi	79.2	tst2013	**31.97**	**6.98**	N/A	**28.33**
			tst2015	**29.22**	**6.44**	N/A	**26.37**
Transformer	IWSLT En-Cz	**126.0**	tst2011	21.88	4.71	23.71	16.48
			tst2013	22.70	5.03	24.18	17.51
MC-GNN	IWSLT En-Cz	123.3	tst2011	**22.25**	**4.95**	**24.81**	**16.95**
			tst2013	**22.99**	**5.23**	**25.14**	**18.10**
Transformer	IWSLT En-De	**148.4**	tst2014	**28.17**	6.05	**43.63**	**24.11**
			tst2015	**30.10**	**6.24**	44.69	**26.05**
MC-GNN	IWSLT En-De	145.8	tst2014	28.15	**6.08**	43.50	23.98
			tst2015	29.42	6.13	**45.35**	25.46

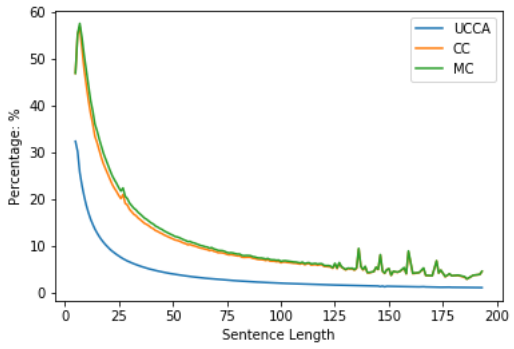


Figure 5: The computational complexity exponentially scales down as sentence length grows.

4.3 Computational Complexity - Experimental Analysis

Since understanding the computational complexity of a model is critical in NLP, we analyze this aspect of the proposed models and the self-attention layer based on sentence length.

The self-attention function connects all tokens in a sentence with themselves, so its complexity is $O(n^2)$, where n is the sentence length. Meanwhile, the attention-based GNN attends each node to their neighbor producing $O(k \cdot n)$ complexity. k is the average degree, and n is the number of nodes or the sentence length. For UCCA graphs, k is significantly smaller than n, and hence $O(k \cdot n)$ is substantially lesser than $O(n^2)$. To give a better illustration, we compare the proportion of k over n in the IWSLT English-to-French datasets to see the decline in real data.

The y-axis of Figure 5 shows the proportion of $O(k \cdot n)$ over $O(n^2)$. We can see that the time complexity of UCCA-GNN is 32% of the Transformer at most and exponentially shrinks to 3%. Meanwhile, the time complexity of CC- and MC-GNN is nearly 60% of the Transformer with a sentence length of 15, and significantly dropped to 10% when the sentence length is longer than 100. Empirically, k is exponentially smaller than n as the sentence length grows, and hence diminishing $O(n^2)$ to $O(k \cdot n)$ enables tasks involving very long sentences.

4.4 Compared with other methods

In this section, we compare our method with FNet (Lee-Thorp et al., 2021), Performer (Choromanski et al., 2021), AFT (Zhai et al., 2021), and Linear Attention (Katharopoulos et al., 2020) in translation tasks. We substitute them with the self-attention

module in the encoder layer, and the experimental results are shown in Table 3.

FNet: is parameter-free with the $O(1)$ time complexity, and thus we double the number of encoder layers for fairness. We find that FNet extremely underfits in translation tasks, while it showed the comparative results against BERT in the encoder-only tasks. We believe that generative tasks are too rigid for Discrete Fourier Transform.

Performer[1]: We set the number of random features to 4 times the head dimension. Although we manually finetune the hyper-parameter settings, the performance is poor on all language-pair datasets.

AFT: We use AFT-full with the $O(T^2d)$ time and $O(Td)$ space complexities. We observe that AFT-full has a better convergence and is more stable when training. The performances are comparative where the drops fluctuate from -0.76 to -2.07 BLEU.

Linear Attention: has the $O(Td^2)$ time complexity, however, the official implementation [2] produced the four-dimensional computation. This makes space complexity spike during the training process, and thus we must use a much smaller batch size - only 16 sentences per batch. Due to the small batch size, Linear Attention is hard to converge and unable to converge in the En-Cz dataset.

Overall, FNet has the fastest training time but the poorest performance among the methods. The AFT method is much more reliable than FNet, Performer, and Linear Attention in translation tasks with stable convergence. Moreover, the results convince us that MC-GNN is more consistent than these works and even outperforms the Transformer in some datasets.

5 Conclusion

MC-GNN gains significant improvements across many translation tasks without changing the architecture, while parameters and the time complexity are reduced. However, there are several limitations in the proposed methods. First, the accuracy of the UCCA parser is below 80%, which can cause some unexpected behaviors and hamper the NMT performance. Second, MC-GNN does not support every language as self-attention, since the UCCA parser is unavailable in every language. Third, we experimented with the graph decoder where a target sentence is treated as a dense graph and built incrementally during the decoding phase (Xu et al., 2021). Although the time complexity of dense graphs and self-attention are $O(T^2d)$, dense graphs are sparse tensor computations, which is extremely memory inefficient and unparallelable. Thus, we dropped this approach due to limited resources.

[1]We use the implementation from https://github.com/lucidrains/performer-pytorch.

[2]https://github.com/idiap/fast-transformers

Acknowledgements

The authors would like to thank Computational Linguistics Center (University of Sciences, HCMC-VNU) for a great support.

We would like to thank the reviewers for their helpful comments.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238, Sofia, Bulgaria. Association for Computational Linguistics.

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2020. Graph convolutional encoders for syntax-aware neural machine translation.

Marzieh Bazrafshan and Daniel Gildea. 2013. Semantic roles for string to tree machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 419–423, Sofia, Bulgaria. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. Lgesql: Line graph enhanced text-to-sql model with mixed local and non-local relations.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2021. Rethinking attention with performers.

Table 3: Experimented on four language-pair datasets.

Model	En-Fr	En-Vi	En-Cz	En-De
Transformer	35.76	28.23	16.48	24.11
FNET	28.35 ↓ 7.41	12.69 ↓ 15.54	11.92 ↓ 4.56	16.90 ↓ 7.21
Performer	28.70 ↓ 7.01	23.26 ↓ 4.97	12.87 ↓ 3.61	20.63 ↓ 3.48
AFT	35.00 ↓ 0.76	27.09 ↓ 1.14	15.14 ↓ 1.34	22.04 ↓ 2.07
Linear Attention	27.64 ↓ 8.12	26.42 ↓ 1.81	N/A	16.15 ↓ 7.96
MC-GNN	36.50 ↑ 0.74	28.33 ↑ 0.10	16.95 ↑ 0.47	23.99 ↓ 0.13

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jonathan L. Gross and Jay Yellen. 2005. *Graph Theory and Its Applications, Second Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-transformer.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. *CoRR*, abs/2006.16236.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *CoRR*, abs/2001.04451.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontañón. 2021. Fnet: Mixing tokens with fourier transforms. *CoRR*, abs/2105.03824.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2021. Fnet: Mixing tokens with fourier transforms.

Diego Marcheggiani, Jasmijn Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 486–492, New Orleans, Louisiana. Association for Computational Linguistics.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. 2021. Random feature attention.

Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 1177–1184, Red Hook, NY, USA. Curran Associates Inc.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2020. Efficient content-based sparse attention with routing transformers. *CoRR*, abs/2003.05997.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks. *6th International Conference on Learning Representations*.

Mingzhou Xu, Liangyou Li, Derek. F. Wong, Qun Liu, and Lidia S. Chao. 2021. Document graph for neural machine translation.

Shuangfei Zhai, Walter Talbott, Nitish Srivastava, Chen Huang, Hanlin Goh, Ruixiang Zhang, and Josh M. Susskind. 2021. An attention free transformer. *CoRR*, abs/2105.14103.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2021. Sparse attention with linear units.

Yanbin Zhao, Lu Chen, Zhi Chen, Ruisheng Cao, Su Zhu, and Kai Yu. 2020. Line graph enhanced AMR-to-text generation with mix-order graph attention networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 732–741, Online. Association for Computational Linguistics.