VISPool: Enhancing Transformer Encoders with Vector Visibility Graph Neural Networks

Tuna Alikaşifoğlu and **Arda Can Aras** and **Aykut Koç**
Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Türkiye
UMRAM, Bilkent University, Ankara, Türkiye
{t.alikasifoglu,can.aras,aykut.koc}@bilkent.edu.tr

Abstract

The emergence of transformers has revolutionized natural language processing (NLP), as evidenced in various NLP tasks. While graph neural networks (GNNs) show recent promise in NLP, they are not standalone replacements for transformers. Rather, recent research explores combining transformers and GNNs. Existing GNN-based approaches rely on static graph construction methods requiring excessive text processing, and most of them are not scalable with the increasing document and word counts. We address these limitations by proposing a novel dynamic graph construction method for text documents based on vector visibility graphs (VVGs) generated from transformer output. Then, we introduce visibility pooler (VISPool), a scalable model architecture that seamlessly integrates VVG convolutional networks into transformer pipelines. We evaluate the proposed model on the General Language Understanding Evaluation (GLUE) benchmark datasets. VISPool outperforms the baselines with less trainable parameters, demonstrating the viability of the visibility-based graph construction method for enhancing transformers with GNNs.[1]

1 Introduction

The advent of transformer-based models has significantly propelled recent advancements in natural language processing (NLP), particularly in natural language understanding (NLU). Transformers, introduced by Vaswani et al. (2017), paved the way for models like GPT (Radford et al., 2019) and BERT (Devlin et al., 2019) and their variants. Due to their ability to capture long-range dependencies and contextual information, these models outperform previous methods in various NLU tasks.

In parallel, graph neural networks (GNNs) have been introduced as potent tools for processing structured data, including graphs in NLP tasks. The GNNs can effectively capture underlying relationships and dependencies by leveraging *message-passing* techniques (Kipf and Welling, 2017; Xu et al., 2019). Notably, Kipf and Welling (2017) introduced graph convolutional networks (GCNs), which have been adapted for text-based graphs in models like TextGCN (Yao et al., 2019) and TensorGCN (Liu et al., 2020), along with other GNN-based methods in NLP (Li et al., 2016; Veličković et al., 2018; Schlichtkrull et al., 2018; Wu et al., 2019; Brody et al., 2022). These methods have shown promise in various NLP tasks by effectively aggregating information from underlying graphs.

However, most GNNs work with a *transductive* structure that combines all training and test instances (nodes) in a single graph and makes predictions for the test nodes. Therefore, adding new training instances or getting predictions on new test instances changes the graph structure, making the previously trained weights obsolete and requiring re-training of the network. Hence, compared to transformer-based approaches, on their own, GNNs do not perform as well in *inductive* learning settings, where the model is required to generalize to unseen data and continuously applied after training. The inherent transductive nature of GNNs also raises an important caution by departing from the traditional assumption of completely unseen test data. These limitations have led to the development of (1) inductive GNNs (Huang et al., 2019; Nikolentzos et al., 2020; Zhang et al., 2020), (2) transductive methods that combine transformers with GNNs (Zhang and Zhang, 2020; Lin et al., 2021; Aras et al., 2024), and finally (3) transformers with inductive GNNs (Huang et al., 2022).

Transductive GNN methods are mostly built on a single large graph structure for the entire corpus with both training and test data, so they are neither applicable to unseen data nor scalable with the increasing document and word counts (Yao et al.,

[1]The source code is available at https://github.com/koc-lab/vispool and all experiment runs can be inspected through WandB dashboard at https://wandb.ai/tunakasif/vispool.

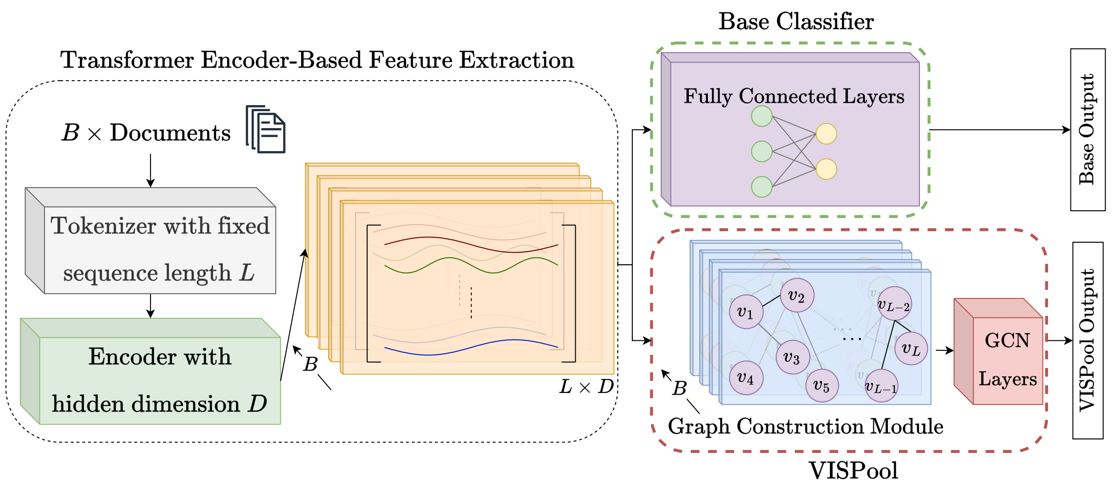


Figure 1: The overall representations of baseline and the proposed method. *The base classifier* (upper path) consists of fully connected layers, which is the current norm for transformer-based classification/regression. The proposed *VISPool* architecture (lower path) that maps document embeddings to graphs and passes to GCN layers.

2019; Lin et al., 2021). Although the current inductive GNN methods construct individual graphs for each document, they rely on static graph construction methods based on statistical metrics such as term frequency-inverse document frequency (TF-IDF), point-wise mutual information (PMI), and co-occurrences (Zhang et al., 2020; Huang et al., 2022). These methods require extensive text preprocessing and are still prone to scalability issues.

In a different vein, visibility graphs (VGs) have been introduced as a novel approach for mapping time series data into a graph (Lacasa et al., 2008; Luque et al., 2009; Bozkurt and Ortega, 2022; Gao and Ge, 2024). Then, Ren and Jin (2019) extends the notion to vector visibility graphs (VVGs) to map multivariate time series. The constructed graphs enable the extraction of sequential relations, facilitating improved analysis and insights. Recent studies introduce GNNs with VG (Xuan et al., 2022; Aslan and Choi, 2024), but to our knowledge, VVG-based GNN has not been proposed before.

In a broader perspective, transformers generate document embeddings as multivariate time series with inherent sequential relations, where VVGs are designed to capture sequential relations. In contrast to statistical methods, VVG preserves sequential topology and can utilize transformer output without text processing. To leverage these capabilities, in this work, we propose (1) a novel approach to generate dynamic graphs per document by constructing VVGs from transformer representations and (2) a novel scalable architecture, *visibility pooler (VISPool)*, that seamlessly integrates VVG-based GCNs into transformer pipelines and enhances performance on General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018).

2 Method

We first briefly outline the mainstream approach and then introduce our novel model for enhancing transformer encoders with VVG-based GCNs, where both architectures are shown in Fig. 1.

2.1 Transformer Encoder and Base Classifier

For a given corpus batch of B documents, a tokenizer with a fixed sequence length of L, and a pre-trained transformer encoder with D-dimensional hidden output, the transformer encoder generates a $B \times L \times D$ dimensional tensor representation for the document batch. The representation contains B sequences of L-length and D-dimensional embeddings. The mainstream approach uses these embeddings with a classifier or regressor head, implemented as fully connected layers. Then, fine-tune the whole network for final predictions on the NLU task. We refer to this approach as *base classifier* and represent it as the *upper path* in Fig. 1.

2.2 Our Approach: VISPool

We introduce VVG-based dynamic graph construction in Section 2.2.1 and a GCN-based model architecture in Section 2.2.2. We refer to the overall as *VISPool* and represent it as the *lower path* in Fig. 1.

2.2.1 Graph Construction Module

Broadly, the transformer encoders generate an embedding matrix of size $L \times D$ for a single document. Due to the structure's intrinsic sequential nature, each document embedding is a D-dimensional, L-length multivariate time series. This section describes how we leverage this structure with VVGs to construct dynamic graphs for each document.

Visibility Graph For an N-length time series $\{(t_i, x_i)\}_{i=0}^{N-1}$, with values $x_i \in \mathbb{R}$ sampled at time instance $t_i \in \mathbb{R}$, the VG is introduced as a mapping from (t_i, x_i) instance to corresponding graph node v_i (Lacasa et al., 2008). The graph edges are determined based on a selected visibility criterion, so the unweighted adjacency matrix entries are:

$$\mathbf{A}_{i,j} = \begin{cases} 1 & \text{if nodes } v_i \text{ and } v_j \text{ are } \textit{visible}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The adjacency can also be weighted based on a metric, e.g., the absolute value difference, inverse time difference, etc. Let (t_i, x_i) and (t_j, x_j) be arbitrary endpoints, with in-between instances (t_k, x_k) such that $i < k < j$. The nodes v_i and v_j are connected, i.e., *visible*, if the visibility criterion is satisfied for all in-between points. A visibility criterion is defined by *line-of-sight* principle where the two most common ones, *natural* (Lacasa et al., 2008) and *horizontal* (Luque et al., 2009), are defined as Eqs. (2) and (3), respectively:

$$\frac{x_j - x_i}{t_j - t_i} < \frac{x_j - x_k}{t_j - t_k}, \tag{2}$$

$$x_k < \min(x_i, x_j). \tag{3}$$

Then, $\mathbf{A}$ is defined in a *left-to-right* directed manner, i.e., for $i < j$, so it is upper triangular. However, the graph can be converted to an undirected graph by equating $\mathbf{A}_{j,i}$ to $\mathbf{A}_{i,j}$.

Limited Penetrability The described visibility criteria are strict and require all in-between values to satisfy the criterion. However, Ting-Ting et al. (2012) show relaxing the criterion and allowing a K number of violations increases robustness to noise and is referred to as *limited penetrability*.

Vector Visibility Graph The VVG generalizes the VG to map a multivariate time series to a graph (Ren and Jin, 2019). For an M-dimensional, N-length multivariate time series $\{(t_i, \mathbf{x}_i)\}_{i=0}^{N-1}$, the VVG is constructed by mapping each vector $\mathbf{x}_i \in \mathbb{R}^M$ at time instance $t_i \in \mathbb{R}$ to a corresponding node v_i. The projection magnitudes are used since the vectors are not directly comparable in the visibility criteria. The projection magnitude of a vector $\mathbf{x}_j$ onto $\mathbf{x}_i$ is defined as:

$$\|\mathbf{x}_j^i\|_2 \triangleq \|\text{proj}_{\mathbf{x}_i}(\mathbf{x}_j)\|_2 = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\|\mathbf{x}_i\|_2}. \tag{4}$$

Then, all value comparisons in the criteria are replaced with the corresponding projection magnitudes. Both visibility criteria and limited penetrability can be directly generalized to VVGs. The vector visibility criteria are generalized for natural and horizontal as Eqs. (5) and (6), respectively:

$$\frac{\|\mathbf{x}_j^i\|_2 - \|\mathbf{x}_i^i\|_2}{t_j - t_i} < \frac{\|\mathbf{x}_j^i\|_2 - \|\mathbf{x}_k^i\|_2}{t_j - t_k}, \tag{5}$$

$$\|\mathbf{x}_k^i\|_2 < \min(\|\mathbf{x}_i^i\|_2, \|\mathbf{x}_j^i\|_2). \tag{6}$$

In this context, each document embedding is represented as a multivariate time series in the form of $\{(t_i, \mathbf{x}_i)\}_{i=0}^{L-1}$ with token embeddings $\mathbf{x}_i \in \mathbb{R}^D$ and equally spaced time instances $t_i = i$. Therefore, we propose mapping each of the B document embeddings at the transformer output to a VVG with selected criteria and limited penetrability, resulting in B number of $\mathbf{A} \in \mathbb{R}^{L \times L}$. Then, these graphs are ready to be used in GCN layers. Since the embeddings are updated with training, the graphs are dynamic and updated accordingly.

2.2.2 Graph Convolutional Network Layers

We directly feed the generated VVG adjacencies $\mathbf{A} \in \mathbb{R}^{L \times L}$ to GCN layers (Kipf and Welling, 2017), since GCNs are the most widely used GNN structures. The ℓ^{th}-layer input-output node embedding relation of the GCN is defined as follows:

$$\mathbf{H}^{(\ell+1)} = \phi(\mathbf{A}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}), \tag{7}$$

where ϕ is an activation function, $\mathbf{H}^{(\ell)} \in \mathbb{R}^{L \times d_\ell}$ is the ℓ^{th}-layer d_ℓ-dimensional node embeddings, and $\mathbf{W}^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$ is a trainable weight matrix. Note that $\mathbf{H}^{(0)}$ is the initial node embeddings and can be set based on the application at hand, e.g., identity matrix, GloVe embeddings (Pennington et al., 2014), etc. We use the transformer encoder document embeddings, i.e., the output of the transformer, as the initial node embeddings $\mathbf{H}^{(0)} \in \mathbb{R}^{L \times D}$ in Eq. (7). Note that we propose a batched version of the GCN, which means training a single weight matrix $\mathbf{W}^{(\ell)}$ for the ℓ^{th}-layer on all B graphs in the batch.

Model	AVG	Single Sequence		Similarity and Paraphrase			Natural Language Inference			
		CoLA	SST-2	MRPC Acc/F1	QQP Acc/F1	STS-B P/SP	MNLI m/mm	QNLI	RTE	WNLI
DistilBERT										
Base Classifier	79.44	55.54	91.63	85.78/89.78	90.39/86.61	82.23/80.01	81.83/82.32	89.62	60.65	**56.34**
VISPool-NVVG	79.07	53.87	90.14	83.58/88.74	90.51/86.83	82.22/80.17	**82.44**/81.88	88.43	62.82	**56.34**
VISPool-HVVG	78.97	53.67	90.48	84.07/88.86	89.44/85.54	**82.56/80.56**	81.90/82.54	89.62	61.01	**56.34**
VISPool-LP-NVVG	79.76	54.77	91.40	**86.27/90.23**	89.76/85.67	82.39/80.26	82.39/**82.71**	89.04	**65.70**	**56.34**
VISPool-LP-HVVG	**79.89**	**57.09**	**91.74**	85.78/89.98	**90.52/86.93**	82.42/80.20	82.08/82.41	**89.93**	63.18	**56.34**
ALBERT										
Base Classifier	82.07	53.77	91.17	88.97/91.76	90.28/86.48	**87.67/85.54**	84.01/84.47	90.24	76.17	**56.34**
VISPool-NVVG	81.73	55.70	92.89	89.71/92.54	90.32/86.48	85.60/83.12	83.93/81.81	90.43	73.65	**56.34**
VISPool-HVVG	81.83	57.54	88.76	89.71/92.52	89.82/85.81	85.87/83.57	84.23/83.11	90.40	76.17	**56.34**
VISPool-LP-NVVG	82.60	**60.45**	93.00	**89.95/92.81**	**90.57/86.86**	85.52/82.82	**84.34/84.88**	90.85	75.45	**56.34**
VISPool-LP-HVVG	**82.74**	59.41	**93.12**	89.46/92.48	90.55/86.80	86.23/83.56	84.25/84.54	**91.21**	**77.62**	**56.34**

Table 1: Maximum achieved scores on the GLUE validation (dev) sets (scaled by 100). We report the corresponding metric for each task (1) Matthews correlation for CoLA (2) Pearson/Spearman correlation for STS-B (3) accuracy/F1 for MRPC and QQP (4) accuracy for all other tasks, where, for MNLI, both on the matched and mismatched sets.

Summary Instead of directly passing the document embeddings to the classifier, we propose to map them to graphs and pass them through GCN layers. On top of the transformer's contextual information, our motivation is to capture relational information with GCNs and ultimately enhance the transformer's performance. For mapping, we propose a VVG-based dynamic graph construction that does not require text processing. The method is also scalable since the graph sizes are only dependent on the selected tokenizer sequence length L, which is usually chosen as $\{128, 256, 512\}$.

3 Experiments

We evaluate our approach by addressing whether, given a pre-trained transformer, the VVG-based GCNs can enhance the performance in NLU compared to the base classifier. Due to their lightweight but competitive performance, we choose DistilBERT (Sanh et al., 2019) and ALBERT (Lan et al., 2019) as base transformers and use their corresponding tokenizers, both provided by *Hugging Face*[2]. Up to this point, the process is identical for the base classifier and VISPool. Then, for the base classifier, we use *SequenceClassifier*[2] head.

On the other hand, for the VISPool path, we implement a fast and parallel VVG generation process, along with a batched GCN layer implementation that extends the notion in *PyTorch Geometric* (Fey and Lenssen, 2019), to minimize the computational overhead, resulting in similar training times with the base classifier. Further details of the experimental setup, including model details and computational infrastructure, are provided in Appendix B.

3.1 Datasets

We conduct experiments on the GLUE (Wang et al., 2018) datasets, which have become the standard benchmark for NLU assessment, where the individual dataset information is detailed in Appendix A.

3.2 Hyperparameter Tuning

For the base classifier, we conduct an individual grid search on each dataset with five seeds and four learning rates suggested by Devlin et al. (2019). We report the scores on the best-performing overall seed and best learning rate for each dataset. For fairness, we use the selected seed for VISPool and employ Bayesian hyperparameter optimization (Dewancker et al., 2015). The list of hyperparameters and their distributions are detailed in Appendix B.4.

3.3 Results

We use DistilBERT and ALBERT transformer models and compare the base classifier with unweighted and undirected VVG-based VISPool variants with natural (NVVG) and horizontal (HVVG) visibility criteria described in Section 2.2.1. We also provide the limited penetrable (LP) variants with $K \in \{1, \ldots, 5\}$ allowed violations (K is treated as a hyperparameter). Since test labels are not publicly available, as a common practice, we report the maximum achieved scores on the validation (dev) sets in Table 1 where the best scores are emboldened.

[2]We use models from huggingface.co/distilbert/distilbert-base-uncased and huggingface.co/albert/albert-base-v2.

On the GLUE benchmark, with fewer trainable parameters, the VISPool-LP variants systematically outperform baselines, improving the overall average score. On the other hand, even though the strict versions' overall average is not better, they still provide improvements in some datasets; there are even cases where they are top performers.

Since relaxing the visibility criteria with limited penetrability results in denser graphs, i.e., graphs with more connections, it can be said that VISPool variants usually perform better with less sparse graph representations. We highlight that the only differences between the variants are the parameters for the VVG graph they generate. Therefore, as the results in Table 1 suggest, the performance of VISPool is highly dependent on the quality of the graph construction, which is adjusted by the hyperparameters of visibility criteria (natural or horizontal) and the limited penetrability K. With the hyperparameter space we cover, VISPool provides performance improvements over base classifiers up to 13% for certain tasks in GLUE benchmarks, resulting in a promising approach for enhancing transformer performance.

4 Conclusion

In this work, we demonstrate both the viability of the VVG approach for graph construction and the effectiveness of the proposed VISPool architecture for enhancing transformer encoders with GNNs in NLU tasks. VISPool is shown to be successful in aggregating relational information that contextual transformers cannot capture. Therefore, we believe that the VVG-based dynamic graph construction is a promising approach for NLP tasks thanks to not needing excessive text processing and being more scalable than existing approaches. The proposed VISPool architecture can be further explored and extended by weighted VVGs and applications to other inductive NLP tasks that can benefit from transformer enhancements with GNNs.

Limitations

Even though the variants of VISPool outperform the baseline models, the performance is affected by the choice of VVG type, i.e., the visibility criterion and the number of allowed violations (penetrable limit). In other words, the model performance depends on the quality of the VVG construction and the transformer encoder output used as the initial node embeddings. Therefore, the model may not perform well on tasks where the transformer encoder output is insufficient to capture the initial contextual information within the document or the VVG construction is inadequate to capture the relational information. Furthermore, naturally, the VISPool depends on the availability of pre-trained transformer encoders, and in the absence of such, a pre-training step is required.

Acknowledgements

The work of Tuna Alikaşifoğlu and Arda Can Aras is supported by Türk Telekomünikasyon A.Ş. within the framework of the 5G and Beyond Joint Graduate Support Program coordinated by the Information and Communication Technologies Authority. The work of Aykut Koç was supported by the 2023 BAGEP Award of the Science Academy.

References

Arda Can Aras, Tuna Alikaşifoğlu, and Aykut Koç. 2024. Graph receptive transformer encoder for text classification. *IEEE Transactions on Signal and Information Processing over Networks*, 10:347–359.

Hacı İsmail Aslan and Chang Choi. 2024. VisGIN: Visibility graph neural network on one-dimensional data for biometric authentication. *Expert Systems with Applications*, 237:121323.

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second PASCAL recognising textual entailment challenge.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge.

Ecem Bozkurt and Antonio Ortega. 2022. Non-negative kernel graphs for time-varying signals using visibility graphs. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pages 1781–1785.

Shaked Brody, Uri Alon, and Eran Yahav. 2022. How attentive are graph attention networks? In *International Conference on Learning Representations*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Zihang Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. 2017. Quora question pairs.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ian Dewancker, Michael McCourt, and Scott Clark. 2015. Bayesian optimization primer.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the International Workshop on Paraphrasing*.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Meng Gao and Ruijun Ge. 2024. Mapping time series into signed networks via horizontal visibility graph. *Physica A: Statistical Mechanics and its Applications*, 633:129404.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics.

Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2019. Text level graph neural network for text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3444–3450, Hong Kong, China. Association for Computational Linguistics.

Yen-Hao Huang, Yi-Hsin Chen, and Yi-Shin Chen. 2022. ConTextING: Granting document-wise contextual embeddings to graph neural networks for inductive text classification. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1163–1168, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Lucas Lacasa, Bartolo Luque, Fernando Ballesteros, Jordi Luque, and Juan Carlos Nuño. 2008. From time series to complex networks: The visibility graph. *Proceedings of the National Academy of Sciences*, 105(13):4972–4975.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The Winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, volume 46, page 47.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated graph sequence neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. 2021. BertGCN: Transductive text classification by combining GNN and BERT. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1456–1462, Online. Association for Computational Linguistics.

Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. 2020. Tensor graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8409–8416.

B. Luque, L. Lacasa, F. Ballesteros, and J. Luque. 2009. Horizontal visibility graphs: Exact results for random time series. *Phys. Rev. E*, 80:046103.

Giannis Nikolentzos, Antoine Tixier, and Michalis Vazirgiannis. 2020. Message passing attention networks for document understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8544–8551.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*, pages 2383–2392. Association for Computational Linguistics.

Weikai Ren and Ningde Jin. 2019. Vector visibility graph from multivariate time series: a new method for characterizing nonlinear dynamic behavior in two-phase flow. *Nonlinear Dynamics*, 97(4):2547–2556.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, pages 593–607, Berlin, Heidelberg. Springer-Verlag.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642.

Zhou Ting-Ting, Jin Ning-De, Gao Zhong-Ke, and Luo Yue-Bin. 2012. Limited penetrable visibility graph for establishing complex network from time series. *Acta Physica Sinica*, 61(3):030506.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments. *arXiv preprint 1805.12471*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL-HLT*.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *International Conference on Learning Representations*.

Qi Xuan, Jinchao Zhou, Kunfeng Qiu, Zhuangzhi Chen, Dongwei Xu, Shilian Zheng, and Xiaoniu Yang. 2022. Avgnet: Adaptive visibility graph neural network and its application in modulation classification. *IEEE Transactions on Network Science and Engineering*, 9(3):1516–1526.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7370–7377.

Haopeng Zhang and Jiawei Zhang. 2020. Text graph transformer for document classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8322–8327, Online. Association for Computational Linguistics.

Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. 2020. Every document owns its structure: Inductive text classification via graph neural networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 334–339, Online. Association for Computational Linguistics.

A Detailed Description of GLUE Datasets

We obtain the GLUE datasets from the *Hugging Face* `datasets` library[3] and use the provided train and validation (dev) splits. We describe the individual datasets below, then give the training and validation splits, evaluation metrics, and the number of labels in Table 2.

CoLA The Corpus of Linguistic Acceptability (Warstadt et al., 2018) is a binary classification task for single sentences that aims to classify whether an English sentence is linguistically acceptable correctly.

SST-2 The Stanford Tree Sentimentbank (Socher et al., 2013) is a binary classification task for single sentences based on film reviews containing human sentiment annotation.

MRPC The Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005) consists of sentence pairs extracted from online news sources with human annotations. A similarity and paraphrase classification task classifies whether the sentences in the pair are semantically equivalent.

[3]https://huggingface.co/datasets/glue

QQP Quora Question Pairs (Chen et al., 2017) is a binary classification task to determine whether two questions asked on Quora are semantically equivalent.

STS-B The Semantic Textual Similarity Benchmark (Cer et al., 2017) consists of sentence pairs drawn from news headlines and other sources. Compared to other tasks in GLUE, STS-B is a regression task since the pairs are annotated with a score from 1 to 5, denoting the similarity of two sentences in terms of semantic meaning.

MNLI Multi-Genre Natural Language Inference is a large-scale, crowdsourced entailment classification task consisting of sentence pairs (Williams et al., 2018). The aim is to predict the relation between the pair, whether the sentences are an entailment, contradiction, or neutral.

QNLI The Question Natural Language Inference is a binary classification conversion of the Stanford Question Answering Dataset (Rajpurkar et al., 2016) by Wang et al. (2018).

RTE The Recognizing Textual Entailment (RTE) datasets come from a series of annual textual entailment challenges by combining the data from RTE1 (Dagan et al., 2006), RTE2 (Bar Haim et al., 2006), RTE3 (Giampiccolo et al., 2007), and RTE5 (Bentivogli et al., 2009).

WNLI The Winograd Schema Challenge is a reading comprehension task (Levesque et al., 2011), although the GLUE webpage indicates issues with the dataset. We obtained the same result for all models, consistent with the reported results in the original DistilBERT approach (Sanh et al., 2019).

Dataset	Evaluation Metric	Train Split	Validation Split	Number of Labels
CoLA	Matthews Corr.	8,551	1,043	2
SST-2	Accuracy	67,349	872	2
MRPC	Accuracy/F1	3668	408	2
QQP	Accuracy/F1	363,846	40,430	2
STS-B	Pearson/Spearman Corr.	5,749	1,500	1
MNLI	Accuracy	392,702	9,815/9,832	3
QNLI	Accuracy	104,743	5,463	2
RTE	Accuracy	2,490	277	2
WNLI	Accuracy	635	71	2

Table 2: Number of documents in training and validation splits, evaluation metric, and number of labels of the GLUE datasets. The STS-B is a regression task with a single label; the remaining are classification tasks. For the MNLI task, both matched and mismatched validation splits are provided M/MM.

B Model and Experimental Setup Details

B.1 Model Naming

The BERT (Devlin et al., 2019) architecture contains a fully connected `BertPooler` layer that takes the output representation corresponding to the first token and uses it for downstream tasks. Since we propose an alternative vector visibility graph-based batched GCN architecture, we name our model as *VISPool*.

B.2 Model Details

The DistilBERT and ALBERT transformer models contain 66.4 and 11.7 million trainable parameters, respectively. On top of this structure, the base classifier (embraced as *SequenceClassifier* from *Hugging Face*) adds two fully connected layers (1) a *pre-classifier* of both input and output dimensions of 768 and (2) a classifier with input dimension 768 and output dimension matching the number of labels in the dataset, which is in $\{1, 2, 3\}$ for GLUE datasets as provided in Table 2.

On the other hand, in VISPool architecture, we replace these two fully connected layers with GCN layers. Since we use the transformer encoder outputs as the initial node embeddings, we have $\mathbf{H}^{(0)} \in \mathbb{R}^{L \times D}$. As we describe in Appendix B.4, we choose $L = 128$ and $D = 786$ due to base language model selection. To have a matching dimensionality in Eq. (7), the first dimension of $\mathbf{W}^{(0)}$ needs to be set to $D = 786$. In this context, we only set a single hidden dimension $H \in \{128, 512\}$ to describe GCN layers: (1) first GCN layer with input dimension of $D = 768$ and output dimension of H and (2) second GCN layer with input dimension of H and output dimension of $\lfloor H/2 \rfloor$. Finally, we add a fully connected layer at the end with the input dimension of $\lfloor H/2 \rfloor$ and the output dimension of the number of labels in the dataset. In GCN layers, we use ReLU activation functions, and for tasks with a number of labels more than one, we add $\log\text{-}\mathrm{SoftMax}$ to match the negative log-likelihood loss. For the STS-B dataset, this is omitted since minimum squared error (MSE) loss is utilized.

With these architectures, on top of the transformer encoder, we provide the number of trainable parameters in the base classifier and the VISPool in Table 3.

We also want to indicate that our source code is publicly available, allowing a more transparent analysis of implementation details.

Model	Trainable Parameter Count
Base Classifier	591,360
VISPool-128 hidden	106,624
VISPool-512 hidden	524,800

Table 3: Number of trainable parameters for different models on datasets with a number of labels 2.

B.3 Computing Infrastructure

We run our experiments on the GPU cluster of our research team. Experiments are mostly run on multiple NVIDIA 1080Ti GPUs, but depending on the availability of the cluster, 2080Ti and 3090 GPUs are also utilized. According to the `run` generation and last heartbeat timestamp logs of `WandB` experiments, we use ~4400 total GPU hours. This duration also includes the GPU hours used in the development process. We publicly share every single experiment run through `WandB` dashboard to be transparent regarding model performance and system usage.

B.4 Hyperparameters

Devlin et al. (2019) suggests learning rates of $\{2 \times 10^{-5}, 3 \times 10^{-5}, 4 \times 10^{-5}, 5 \times 10^{-5}\}$ for the transformer encoder. Therefore, the base classifier's fine-tuning process is set to a grid search between these learning rates and five different seeds of $\{40, 41, 42, 43, 44\}$. Within these seeds, 42 is the top performing in the overall average. Therefore, we conduct a hyperparameter search on VISPool with a single seed of 42. We provide the summary for the list and distributions of the hyperparameters used in experiments in Table 4. We further explain our approach as follows.

As a common practice, we have fixed selections of batch size as 32 and tokenizer length as 128. Also, with the choice of the language model as DistilBERT (base-uncased) and ALBERT (base-v2), we overall have $(B, L, D) = (32, 128, 768)$. Therefore, we only have the transformer encoder's learning rate as a hyperparameter. We choose the learning rate to be uniform in $[10^{-5}, 10^{-4}]$ to encapsulate the suggested values.

On the VVG side, we have a categorical parameter of visibility "type", either natural or horizontal. In addition, penetrable limit K is also treated as a hyperparameter. Technically, K can be as large as the sequence length L, but it would generate a

Hyperparameter	Distribution
Encoder	
learning rate	$\sim \text{Uniform}(10^{-5}, 10^{-4})$
VVG	
type	$\in \{\text{natural}, \text{horizontal}\}$
penetrable limit	$\in \{0, 1, 2, 3, 4, 5\}$
GCN	
learning rate	$\sim \text{Uniform}(10^{-5}, 10^{-2})$
dropout	$\sim \text{Uniform}(0.1, 0.5)$
hidden dimension	$\in \{128, 512\}$
pooling	$\in \{\texttt{[CLS]}, \text{mean}, \text{max}\}$

Table 4: The hyperparameter distributions utilized during the experiments, where the categorical distributions have the same initial sampling probability.

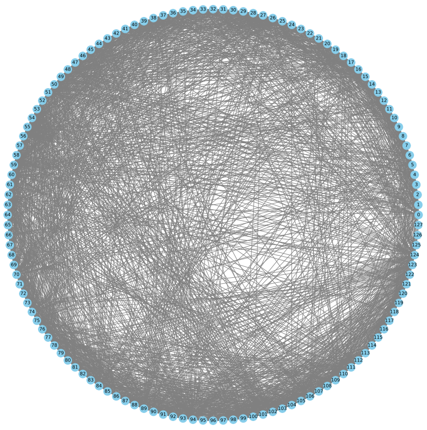
fully connected graph. Since large K can create unnecessary edges in the VVG, we experiment with $K \in \{0, 1, 2, 3, 4, 5\}$ to also decrease the search grid. We highlight that $K = 0$ corresponds to strict visibility criteria.

Finally, on the GCN side, along with the classical hyperparameter selections of learning rate, dropout, and hidden dimension, we also provide an option to select the pooling method at the end. By default, BERT variants pool the `[CLS]` tokens at the end, but due to the provided architecture, we also have the flexibility to choose how to treat the pre-logits. We experiment with classical approaches of still using `[CLS]` token (taking the first embedding), using mean and max approaches.

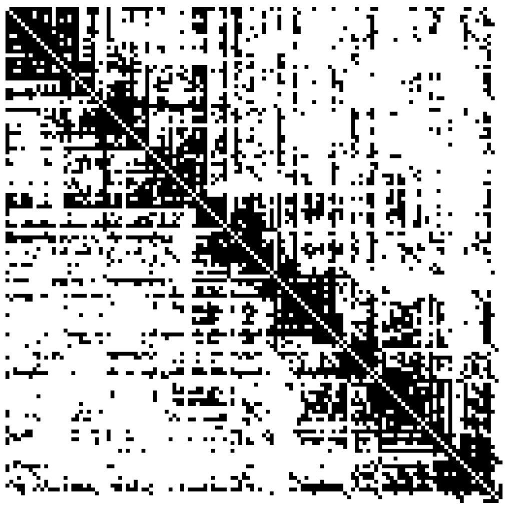
Last but not least, due to common practice, we use Adam with weight decay as an optimizer for both base classifier and VISPool, with default values of *PyTorch* implementation: $(\beta_1, \beta_2) = (0.9, 0.999), \epsilon = 10^{-8}$ and weight decay $\lambda = 0.01$. We publicly share the best-performing hyperparameters and the runs that achieve maximum scores directly with `WandB` dashboard.

B.5 Graph Visualization

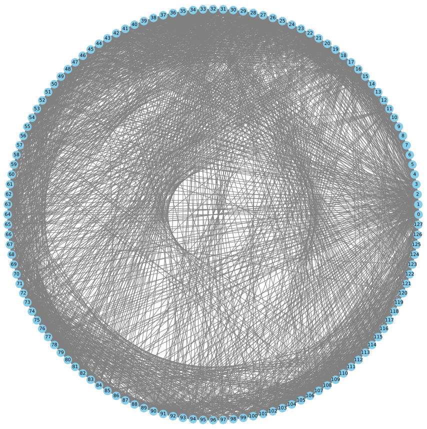
Finally, we demonstrate the gradual change of the VVG representations during the training in Fig. 2. We provide an example for the first document of the RTE dataset with the setting of a DistilBERT transformer and VISPool-NVVG variant. We both provide the graph and the binary representations of the adjacency matrix to provide intuition for the generated graphs, where they visualize the connections between $L = 128$ tokens. We provide `eps` format images and suggest zooming in for details of the figures.



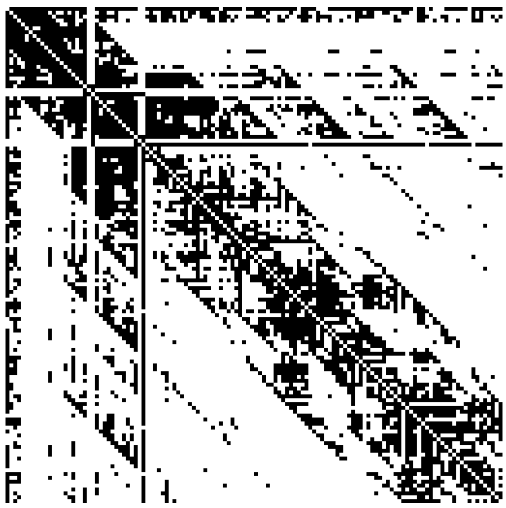
(a) Epoch 1, VVG



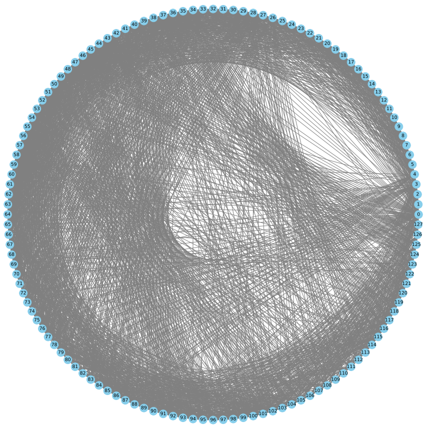
(b) Epoch 1, Binary Adjacency



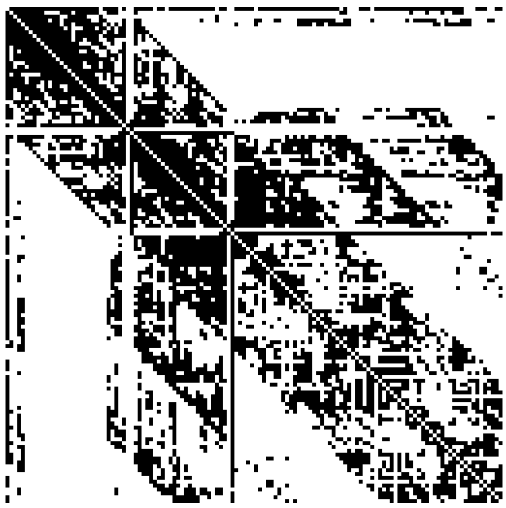
(c) Epoch 5, VVG



(d) Epoch 5, Binary Adjacency



(e) Epoch 10, VVG



(f) Epoch 10, Binary Adjacency

Figure 2: Gradual change of the first document of the RTE dataset with the setting of a DistilBERT transformer and VISPool-NVVG variant. Left-hand side subfigures represent the generated VVG at each epoch, whereas right-hand side subfigures are the binary representation of their adjacency matrices, where black pixels indicate a connection between nodes. For epochs 1, 5, and 10, we obtain validation accuracies of 47.29%, 52.70%, and 62.82%, with graph sparsities of 29.15%, 28.44%, and 33.29%, respectively.