GAT - ATTENTION AND MEMORY

Annisa Haycal
Bina Nusantara University
haycal.annisa@binus.ac.id

Catherine Benedicta
Bina Nusantara University
benedicta.catherine@binus.ac.id

Sekar Azalea
Bina Nusantara University
mardiah.sekar@binus.ac.id

ABSTRACT

Model Graph Neural Network (GNNs) terkenal dengan penggunaannya untuk permasalahan suatu grafik. Dengan menerapkan mekanisme attention untuk menyelesaikan tugas ini, maka digunakanlah arsitektur Graph Attention Network (GATs) karena arsiterktur ini dianggap lebih akurat untuk permasalahan ini dibandingkan model GNN lainnya, karena mekanisme attention dapat menyelesaikan input node dengan ukuran yang berbeda-beda, dan hanya berfokus pada bagian yang penting. Dugaan ini dibuktikan dengan hasil eksperimen kami dengan membandingkan model GAT dengan model GCN. Dengan menggunakan dataset CORA, hasil akurasi yang didapatkan GAT lebih tinggi dibandingkan dengan hasil GCN.

***K*eywords** attention, GAT, node classification

1 Introduction

Permasalahan *node classification* merupakan tugas dimana algoritma melakukan pengenalan mengenai label dari node-node tetangganya. Tugas ini memprediksi atribut dari setiap node dalam grafik. Misalnya, memberi label pada setiap node dengan kelas kategori (klasifikasi biner atau klasifikasi multi kelas), atau memprediksi angka kontinu (regresi). Tugas ini juga dilakukan untuk memprediksi node yang tidak ada, yang dapat dikenali berdasarkan properti node lainnya.

Model seperti Convolutional Neural Networks (CNNs) dan Recurrent Neural Networks (RNNs) umumnya dapat menyelesaikan permasalahan seperti *node classification*. Namun, tantangan pada pendekatan ini adalah dimana algoritma harus menentukan operator yang bekerja dengan node tetangga dengan properti yang berbeda. Pada beberapa kasus, hal ini membutuhkan pembelajaran mesin mengenai *matrix weight* pada masing-masing node.

Mekanisme Attention telah menjadi suatu standar de facto pada beberapa masalah dengan *many sequence-based*. Dan salah satu kelebihan dari menggunakan mekanisme adalah algoritma ini dapat menyelesaikan input node dengan ukuran yang berbeda-beda, dan hanya berfokus pada bagian yang penting. Dibandingkan dengan model CNNs dan RNNs, mekanisme ini sudah terbukti lebih berguna untuk masalah seperti machine reading. Tidak hanya itu, pada penelitian yang dilakukan Vaswani[1] pada tahun 2017 menunjukkan bahwa mekanisme juga cukup untuk membangun model terbaik yang memperoleh kinerja canggih dalam tugas machine translation.

Berdasarkan penelitian ini, kita gunakan arsitektur berdasar mekanisme Attention untuk menyelesaikan tugas *node classification* dari grafik terstruktur. Intinya adalah untuk menghitung representasi tersembunyi dari setiap node dalam grafik, dengan memperhatikan tetangganya. Arsitektur ini disebut dengan Graph Attention Networks atau GATs. Berdasarkan kutipan yang diambil dari beberapa artikel mengenai GATs, arsitektur ini merupakan salah satu arsitektur dari model GNNs (Zhang[2]). Maka dari itu, konsep dari arsitektur ini hampir sama dengan arsitektur buatan GNNs lainnya, yaitu GCNs (Labonne[3]).

2 Previous Research

Permasalahan *node classification* sudah umum dilakukan. Permasalahan ini dapat diselesaikan dengan banyak cara dengan banyak model juga. Model CNNs dapat digunakan untuk menyelesaikan permasalahan *node classification*. Walaupun pada dasarnya model ini bukan diperuntukkan untuk masalah grafik seperti yang ada di *node classification*,

namun dengan menggeneralisasikan model ini kepada grafik akan bisa memberikan kemungkinan untuk mencapainya. Penelitian yang pernah dilakukan sebelumnya pada tahun 2015 oleh Duvenaud[4] dan lainnya membuat model CNNs yang beroperasi secara langsung terhadap grafik. Lalu, terdapat juga penelitian Atwood dan Tosley[5] dan Hamilton[6] yang mengambil konsep sama, yaitu untuk mengoperasikan model CNNs ke suatu grafik. Selain itu, penelitian yang dilakukan oleh Monti[7] dan lainnya dengan membuat suatu model campuran CNN dengan nama MoNet juga dilakukan untuk mengoperasikan model CNN ke dalam grafik.

Namun, keempat penelitian tersebut memberikan hasil kesimpulan bahwa untuk mengoperasikan model CNNs ke suatu grafik masih harus membutuhkan beberapa syarat tambahan.

Berhubungan dengan model yang akan kami buat, penelitian yang telah dilakukan Santoro[8] dan Schlichtkrull[9] memiliki pendekatan yang hampir sama dengan kami, dimana pada penelitian ini membahas mengenai *relational networks* akan berbagi informasi antar-node nya. Penelitian kami juga memiliki kesamaan konsep model *attention* dengan penelitian yang dilakukan Duan[10] dan Denil[11] dimana memanfaatkan mekanisme attention di sekeliling node untuk menghitung koefisien attention antara node atau objek di lingkungannya.

Pendekatan yang sama juga pernah dilakukan pada penelitian yang dilakukan oleh Roweis dan Raul[12] yang membuat model LLE (Locally Linear Embedding) dan Weston[13] yang mengangkat topik mengenai memory networks. Model LLE akan menggunakan angka pasti dari tiap node tetangga dan mengumpulkan properti node-node tersebut, seperti koefisien bobot dari tiap nodenya. Dan dengan *memory networks* menganggap tiap node di lingkungan sebagai sebuah memori yang menyimpan properti atau nilai. Dengan inilah kita bisa menghitung fitur-fiturnya dengan memperhatikan nilainya, dan kemudian informasi tersebut akan diperbarui dengan menyimpan fitur baru pada lokasi node yang sama.

Model dari metode GNNs atau Graph Neural Network yang bernama DEMO-Net juga pernah dilakukan oleh Wu[14]. Pada penelitian ini, diusulkan konsep Multi-task Graph Convolution dimana tiap node akan direpresentasikan dengan tingkat yang berbeda-beda. Konsep ini tentunya sama dengan konsep *attention* yang akan kita gunakan. Masih menggunakan metode GNN yang sama, penelitian yang pernah dilakukan oleh Fey[15] dan Maurya[16] meneliti untuk menyeleksi node yang akan dipakai. Berhubungan dengan kedua penelitian tersebut, Maurya[17] juga meneliti bahwa berdasarkan properti yang dimiliki setiap node, peran yang akan dimainkan juga akan berbeda. Dengan mengetahui properti dari tiap node, akan membuahkan hasil yang kemungkinan dapat meningkatkan hasil dari eksperimen. Kipf[18] juga meneliti sebuah representasi yang tersembunyi dari sebuah fitur node. Sebelumnya juga terdapat penelitian oleh Hussain[19] yang mengangkat topik *self-attention* pada GNNs dimana mekanisme *attention* digunakan untuk melihat apakah terdapat interaksi antar-node pada lingkungannya saja (*local self-attention*) atau seluruh input pada grafik (*global self-attention*).

Terinspirasi dari penelitian-penelitian tersebut, penelitian kali ini mengangkat topik mekanisme *attention* dengan arsitektur GAT atau Graph Attention Network. Penelitian yang pernah dilakukan sebelumnya yang juga menggunakan konsep seperti ini terbagi menjadi dua jenis pembelajaran, yaitu *transductive* dan *inductive*. Jenis *transductive* pernah dilakukan oleh Sen[20] dan Yang[21], sedangkan jenis *inductive* dilakukan oleh Zitnik[22], Hamilton[6], dan Subramanian[23].

Beberapa perbedaan penelitian yang pernah dilakukan dengan penelitian yang dilakukan pada paper ini dijelaskan secara singkat pada:

Judul	**Peneliti**	**Metode**	**Hasil**	**Perbedaan**
Semi-Supervised Classification With Graph Convolutional Networks[18]	**Thomas N. Kipf dan Max Welling**	**Mereka membuat struktur grafik menggunakan model *neural network* dan men-*train supervised target* untuk semua node dengan label. Penyelesaian masalah dengan berdasarkan metode GCN.**	**Model GCN yang diusulkan berhasil untuk meng-*encode* struktur grafik dengan fitur node yang berguna untuk usahanya dalam percobaan *semi-supervised classification***	**Metode yang digunakan masih membutuhkan beberapa ketentuan, seperti eigendecompositions dan operasi matriks tentang bobot dari node. Model kami secara implisit memungkinkan untuk langsung menentukan mana node yang lebih penting untuk dikunjungi.**

Watch Your Step: Learning Node Embeddings via Graph Attention[24]	**Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, Alex Alemi**	**Dalam makalah ini, mereka mengganti *hyperparameter* dengan yang dapat dilatih yang secara otomatis berdasarkan *backpropagation***	**Akurasi yang dihasilkan pada model paper ini sebesar 67.9**	**Model GAT pada paper ini menggunakan *hyperparameter* lain selain yang digunakan pada model GAT kita**
Inductive Representation Learning on Large Graphs[6]	**William L. Hamilton, Rex Ying, Jure Leskovec**	**Paper ini menyajikan model GraphSAGE, yang mana model ini merupakan kerangka kerja induktif umum yang memanfaatkan informasi fitur node untuk menghasilkan *node embedding* secara efisien untuk data yang sebelumnya tidak terlihat**	**Pada model LSTM dari GraphSage (GraphSage-LSTM) menghasilkan akurasi 0.612**	**Implementasi dari GraphSAGE dan model GAT kita sangat berbeda, GraphSAGE berbasis sampel, di mana tetangga dari sebuah node dijadikan sampel sebagai nomor tetap, sementara GAT mempertimbangkan semua tetangga**
Representation Learning on Graphs with Jumping Knowledge Networks[25]	**Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe**	**Metode yang digunakan pada paper ini adalah membuat skema agregasi baru untuk pembelajaran representasi node yang dapat mengadaptasi rentang lingkungan ke node secara individual. JKNet ini dapat meningkatkan representasi khususnya untuk grafik yang memiliki subgraf dengan struktur lokal yang beragam, dan karenanya mungkin tidak ditangkap dengan baik oleh agregasi lingkungan dalam jumlah tetap**	**Kesimpulan dari hasil penelitian yang dilakukan pada paper ini adalah bahwa model JKNets ini dapat meningkatkan representasi khususnya untuk grafik yang memiliki subgraf dengan struktur lokal yang beragam, dan karenanya mungkin tidak ditangkap dengan baik oleh agregasi lingkungan dalam jumlah tetap**	**Model yang kita pakai, yaitu GAT mencapai akurasi terbaiknya hanya dengan 2 atau 3 *layer*, menunjukkan bahwa informasi lokal merupakan sinyal yang lebih kuat untuk klasifikasi daripada informasi global. Namun, JKNet mencapai akurasi terbaiknya dengan 6 *layer* menunjukkan bahwa informasi global bersama dengan lokal akan membantu meningkatkan kinerja**
Revisiting Semi-Supervised Learning with Graph Embeddings[21]	**Zhilin Yang, William W. Cohen, Ruslan Salakhutdinov**	**Paper ini menggunakan model bernama Planetoid. Metode yang dipakai yaitu melatih (*train*) *node embedding* di tiap instansi untuk bersama-sama memprediksi label kelas dan konteks lingkungan dalam grafik**	**Angka akurasi yang dihasilkan dengan metode ini adalah 75.7**	**Model Planetoid pada paper ini yang digunakan merupakan penggabungan (*embedding*) grafik untuk mengenali properti atau kelas label dari *node neighbor*, sedangkan model GAT menetapkan angka weight yang berbeda di tiap *neighborhood***

3 Methodology or Architecture Deep Learning

3.1 Node Classification

Node Classification atau klasifikasi node adalah sebuah tugas machine learning umum yang diterapkan pada suatu grafik, dimana kita akan men-*train* suatu model untuk mengklasifikasi node. Algoritma akan melakukan pengenalan mengenai label dari suatu node dan tetangganya. Tugas ini dilakukan untuk memprediksi atribut atau properti dari setiap node dalam grafik. Misalnya, *node classification* dilakukan dengan memberi label pada setiap node dengan kelas kategori (klasifikasi biner atau klasifikasi multi kelas), atau memprediksi angka kontinu (regresi). Tugas ini juga dilakukan untuk memprediksi node yang tidak ada, yang dapat dikenali berdasarkan properti node lainnya.

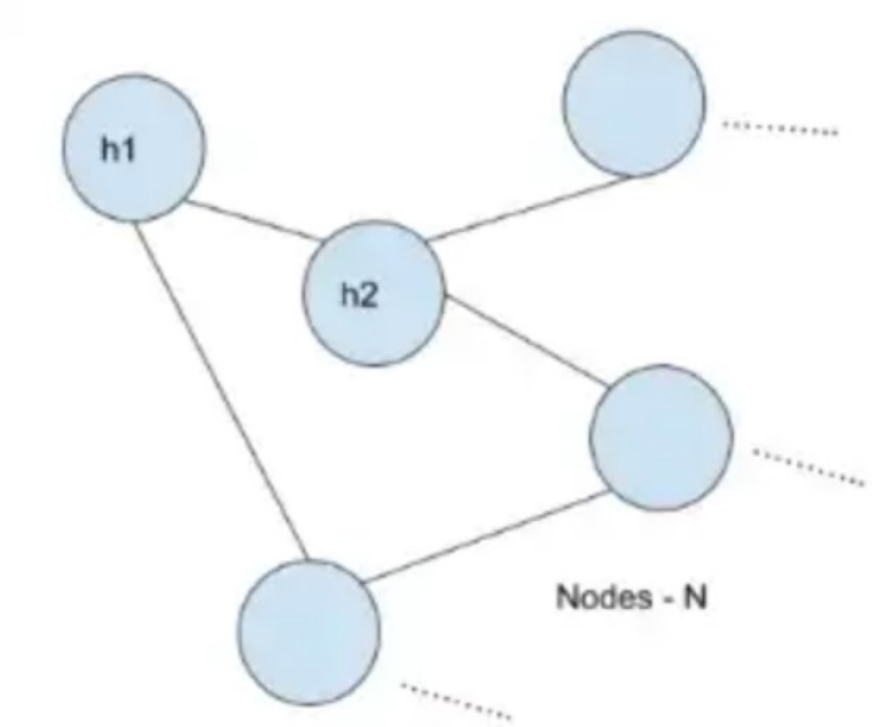
3.2 Graph Neural Networks

GNN atau *graph neural network* sendiri merupakan salah satu arsitektur deep learning yang digunakan untuk menyelesaikan masalah-masalah machine learning pada data yang berbentuk grafik. GNN ini dapat ditinjau sebagai bentuk umum dari Convolutional Neural Network (CNN) dari struktur kisi (piksel dalam gambar) yang cenderung 'kaku' menjadi data yang lebih fleksibel namun tetap terstruktur.

Data pada GNN, jelasnya, berbentuk vektor bilangan riil dengan alur pertukaran informasi yang tidak memiliki aturan ataupun giliran secara spesifik. Vektor awal kemudian mengalami agregasi setelah menerima data dari titik terdekatnya. Setelah adanya proses agregasi, perlu diputuskan hal apa yang ingin dikomputasikan. Terdapat dua pilihan pada GNN, yakni untuk mengkomputasi *node-level function* atau *graph-level function*. GNN sendiri memiliki jumlah layer atau lapis yang telah ditentukan, dengan setiap layer-nya memiliki fungsi agregasi yang berbeda.

3.3 Graph Attention Network (GAT)

Graph Attention Network atau GAT adalah metode pembelajaran non-spektral yang memanfaatkan informasi spasial dari node secara langsung untuk pembelajaran. Ini berbeda dengan pendekatan spektral dari Graph Convolutional Network yang mencerminkan dasar-dasar yang sama dengan Convolutional Neural Net. Bentuk dasar dari GAT adalah Graph Attention Layer. Untuk menjelaskan grafik berikut digunakan sebagai contoh.



An Example Graph

3.3.1 Step 1 : Linear Transformation

Step pertama dalam melakukan Graph Attentional Layer adalah dengan menerapkan transformasi linear. Weight matrix W ke vector dari node.

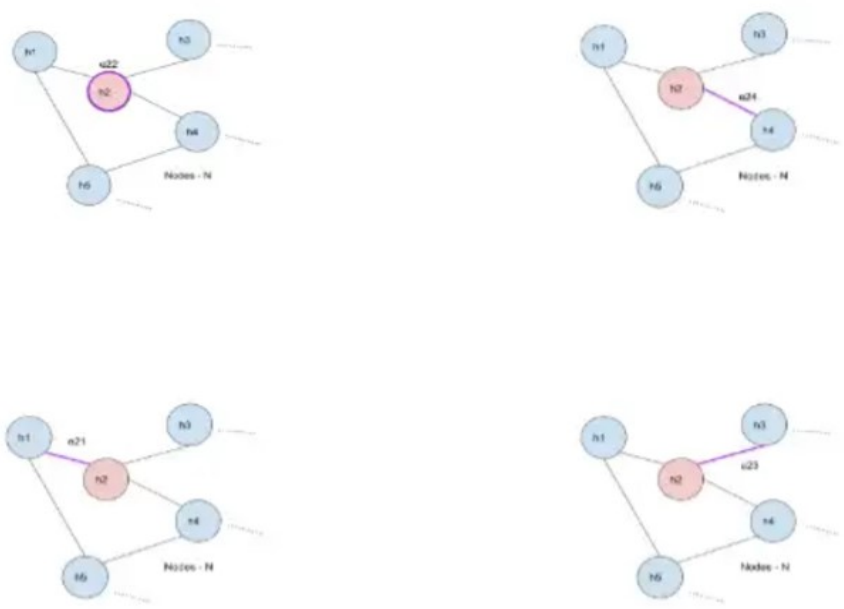
3.3.2 Step 2 : Computation of Attention Coefficients

Koefisien ini menentukan kepentingan dari node neighbornya.

$$e_{ij} = a(W\overrightarrow{h_i}, W\overrightarrow{h_j})$$

dimana i,j adalah node tetangga atau neighbor, dan a adalah fungsi yang ditentukan dari persamaan berikut.

$$a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}.$$



Gambar diatas merupakan visualisasi dari tiap step yang telah kita lakukan.

3.3.3 Step 3 : Normalization of Attention Coefficients

Karena struktur grafik yang bervariasi, node dapat memiliki jumlah neighbor atau tetangga yang berbeda. Untuk memiliki penskalaan umum di semua lingkungan, koefisien attention harus dinormalisasi.

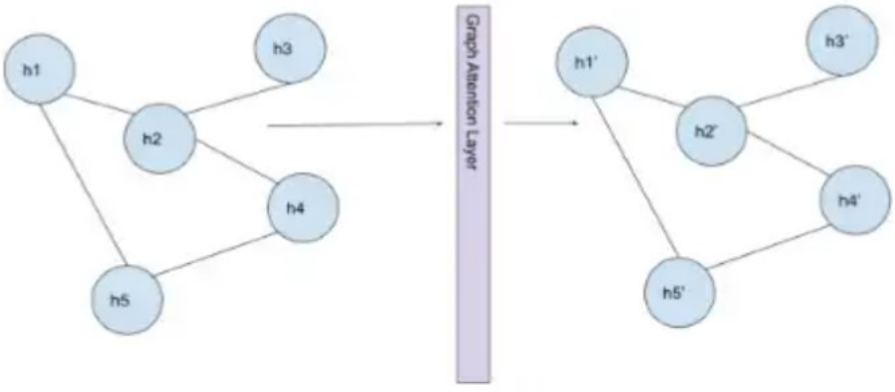
$$\alpha_{ij} = \frac{\exp(LeakyReLU(e_{ij}))}{\sum_{k \in N} \exp(LeakyReLU(e_{ij}))}$$

, dimana N adalah node tetangga dari node-i

3.3.4 Step 4 : Computation of Final Output Features

Sekarang kami menghitung fitur node yang dipelajari. σ adalah Transformasi Non-Linear.

Contoh dari arsitektural network,



Dan formula untuk Learned Output Features,

$$\overrightarrow{h'_i} = \sigma(\sum_{j \in N} \alpha_{ij} W \overrightarrow{h'_j})$$

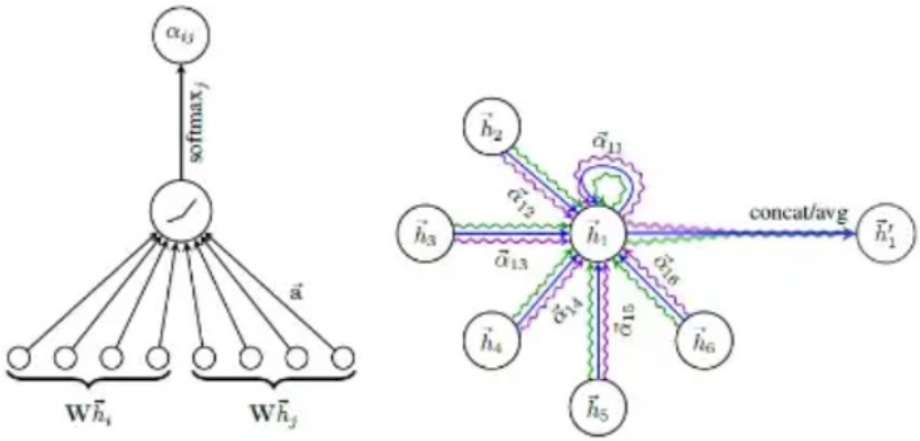
3.3.5 Step 5 : Computation of Multiple Attention Mechanisms

Untuk meningkatkan stabilitas proses pembelajaran, multi-head attention digunakan. Kami menghitung beberapa jalur attentional yang berbeda dan akhirnya mengumpulkan semua representasi yang dipelajari.

$$\overrightarrow{h'_i} = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N} \alpha_{ij}^k W^k \overrightarrow{h'_j})$$

Final Computation of Learned Features

K menunjukkan jumlah jalur attention yang digunakan.



Gambar diatas merupakan tampilan keseluruhan pada semua operasi yang terlibat dalam pembaruan fitur node yang dipelajari.

4 Result and Discussion

Sebelumnya, telah disebutkan bahwa terdapat dua jenis konsep pada model GNN dengan arsitektur GAT, yaitu *transductive* dan *inductive*. Perbedaan antara kedua konsep itu adalah transductive menggunakan dataset yang memiliki satu grafik, seperti dataset Cora. Konsep utama dari jenis ini adalah tidak menggunakan informasi label dari node, melainkan menggunakan informasi seperti struktur dan fitur dari tiap node tersebut. Sedangkan jenis *inductive* menggunakan grafik yang memiliki data terpisah antara *validation, train*, dan *test set*-nya.

Dalam penelitian ini, kami gunakan konsep *transductive* dengan menggunakan dataset CORA. Dataset ini memiliki 2708 node, dimana setiap nodenya memiliki 1433 fitur. Dengan menggunakan dataset ini, kami terapkan pada permasalahan yang telah kami pilih, yaitu *node classification*.

Kita gunakan arsitektur bawaan dari Graph Neural Network (GNN), yaitu Graph Attention Network (GAT) yang menerapkan mekanisme Attention. Konsep dari mekanisme ini adalah mempelajari kekuatan koneksi antar-tetangga node.

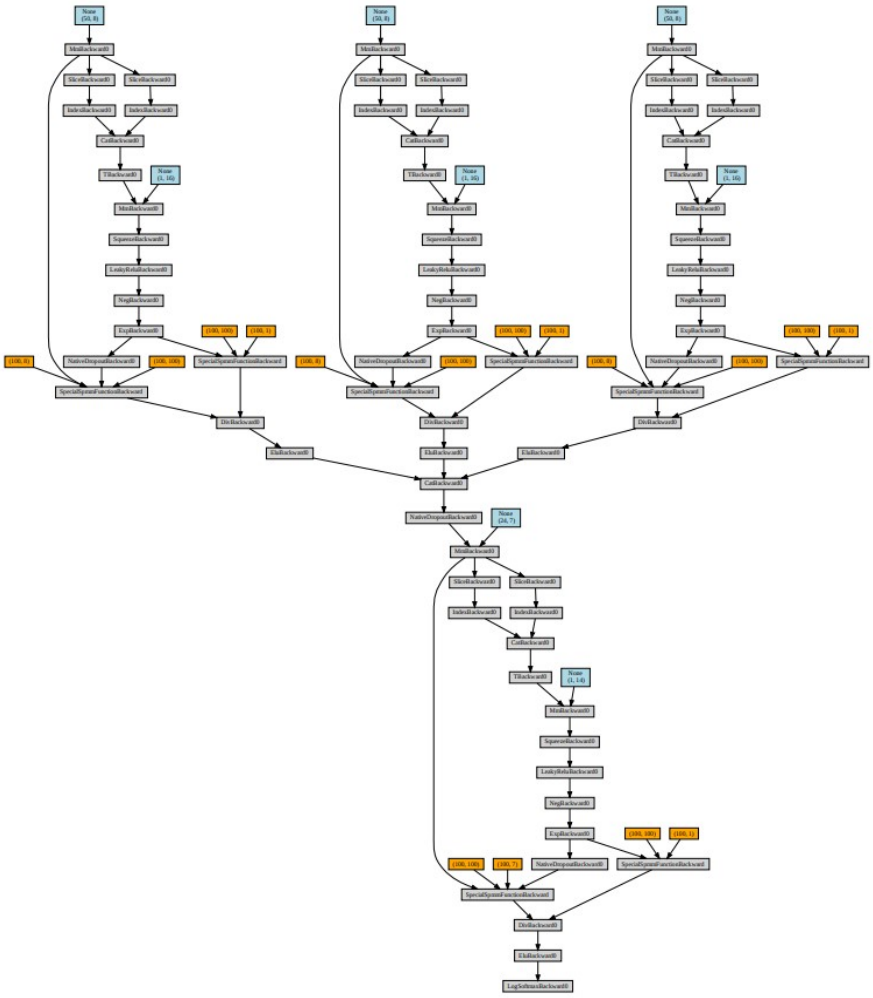
4.1 Training Model GAT

Dari eksperimen dari bentuk implementasi paper rujukan (Velickovic)[26], hasil dari pelatihan data pada model GAT yang kita dapatkan sebesar 0.8410 dengan loss sebesar 0.6626. Angka ini didapatkan dari pengulangan Epochs sebanyak 718 kali.

```
Optimization Finished!
Total time elapsed: 59.5648s
Loading 617th epoch
Test set results: loss= 0.6626 accuracy= 0.8410
```

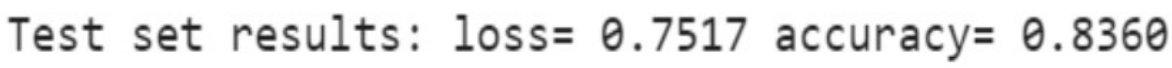
4.2 Visualisasi Model

Berdasarkan implementasi tersebut juga didapatkan visualisasi dari hasil eksperimen yang telah dilakukan.



4.3 Eksperimen antara GCN dan GAT

Kami juga melakukan perbandingan antara dua model arsitektur dari GNN ini, yaitu GCN yang pernah diteliti oleh Kipf[16]. Pada penelitian ini, metode yang digunakan masih membutuhkan beberapa ketentuan, seperti *eigendecompositions* dan operasi matriks tentang bobot dari node. Pada eksperimen dengan menggunakan arsitektur GCN ini, didapatkan hasil akurasinya sebesar 0.836 dengan train loss nya sebesar 0.7517.



5 Conclusion

Pada penelitian kali ini, kita menyelesaikan sebuah permasalahan *node classification*, dimana kita diminta untuk melakukan pengenalan mengenai label dari node-node tetangganya. Tugas ini memprediksi atribut dari setiap node dalam grafik. Kita gunakan arsitektur berdasar mekanisme Attention untuk menyelesaikan tugas ini dari grafik yang ada pada dataset CORA. Dengan mekanisme ini, kita menghitung representasi tersembunyi dari setiap node dalam grafik, dengan memperhatikan tetangganya. Arsitektur ini disebut dengan Graph Attention Networks atau GATs. Lalu, dari arsitektur ini, kita juga lakukan perbandingan hasil yang didapatkan dari arsitektur yang memiliki kemiripan dengan GAT, yaitu GCN. Dari hasil eksperimen yang telah dilakukan, kita dapat melihat bahwa metode GAT memiliki akurasi yang lebih besar dibandingkan dengan metode GCN yaitu sebesar 84.1. Hal ini membuktikan bahwa metode GAT lebih baik digunakan untuk *node classification* dibandingkan dengan metode GCN.

Metode	**Akurasi**
GCN	83.6
GAT	**84.1**

Dengan arsitektur GAT ini, terdapat solusi yang dapat diperoleh, seperti dapat memperluas metode untuk melakukan klasifikasi grafik pada klasifikasi node yang lebih relevan terhadap perspektif aplikasi.

References

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

[2] Hao Zhang, Mufei Li, and Minjie Wang Zheng Zhang. Understand graph attention network. *https://docs.dgl.ai/en/0.8.x/tutorials/models/*$1_{g}nn/9_{g}at.html$, 2018.

[3] Maxime Labonne. Graph attention networks: Self-attention for gnns. *https://mlabonne.github.io/blog/gat/*, 2022.

[4] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gomez-Bombarelli, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. 2015.

[5] James Atwood and Don Towsley. Diffusion-convolutional neural networks. 2015.

[6] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. 2017.

[7] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. 2016.

[8] Adam Santoro, David Raposo, David G.T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. 2017.

[9] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. 2017.

[10] Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. 2017.

[11] Misha Denil, Sergio Gómez Colmenarejo, Serkan Cabi, David Saxton, and Nando de Freitas. Programmable agents. 2017.

[12] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. 1993.

[13] Jason Weston, Frederic Ratle, and Ronan Collobert. Deep learning via semi-supervised embedding. 2008.

[14] Jun Wu, Jingrui He, and Jiejun Xu. Demo-net: Degree-specific graph neural networks for node and graph classification. 2019.

[15] Matthias Fey. Just jump: Dynamic neighborhood aggregation in graph neural networks. 2019.

[16] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Not all neighbors are friendly: Learning to choose hop features to improve node classification. 2022.

[17] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Simplifying approach to node classification in graph neural networks. 2021.

[18] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 2016.

[19] Md Shamim Hussain, Mohammed J. Zaki, and Dharmashankar Subramanian. Global self-attention as a replacement for graph convolution. 2021.

[20] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. 2008.

[21] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. 2016.

[22] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. 2017.

[23] Aravind Subramanian, Pablo Tamayo, Vamsi K. Mootha, Sayan Mukherjee, Benjamin L. Ebert, Michael A. Gillette, Amanda Paulovich, Scott L. Pomeroy, Todd R. Golub, Eric S. Lander, and Jill P. Mesirov. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. 2005.

[24] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alex Alemi. Watch your step: Learning node embeddings via graph attention. 2017.

[25] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. 2018.

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. 2017.