به نام خدا

دانشگاه صنعتي امیرکبیر (پلی تکنیک تهران)

دانشكده مهندسی کامپیوتر



تحلیل شبکه‌های پیچیده

پروژه نهایی

دانشجو: رضا ساجدی

400131072

استاد: دکتر چهرقانی

**پاییز 1401**

فهرست مطالب

[سؤال اول 3](#_Toc126966867)

[قسمت الف 3](#_Toc126966868)

[قسمت ب 4](#_Toc126966869)

[قسمت ج 11](#_Toc126966870)

[سؤال دوم 12](#_Toc126966871)

[قسمت الف 12](#_Toc126966872)

[قسمت ب 14](#_Toc126966873)

[قسمت ج 15](#_Toc126966874)

# سؤال اول

## قسمت الف

این مجموعه‌داده، زیرمجموعه‌ای از وب‌سایت کتابشناسی علوم کامپیوتر DBLP است که شامل یک گراف ناهمگون[[1]](#footnote-2) غیرجهت‌دار با چهار نوع موجودیت (گره) می‌باشد. از این گراف در مسئله طبقه‌بندی گره‌ها استفاده می‌شود و هدف آن یادگیری یک مدل به‌منظور طبقه‌بندی نویسنده‌ها در چهار زمینه پایگاه‌داده، داده‌کاوی، هوش مصنوعی و بازیابی اطلاعات است.

DBLP()

Number of graphs: 1

Number of nodes: 26128

Number of edges: 239566

Number of node features: {'author': 334, 'paper': 4231, 'term': 50, 'conference': 1}

Number of edge features: {('author', 'to', 'paper'): 0, ('paper', 'to', 'author'): 0, ('paper', 'to', 'term'): 0, ('paper', 'to', 'conference'): 0, ('term', 'to', 'paper'): 0, ('conference', 'to', 'paper'): 0}

Number of classes: 4

Average node degree: 9.17

Number of training nodes: 400

Number of validation nodes: 400

Number of test nodes: 3257

Is directed: False

Has isolated nodes: False

Has self loops: False

Other information:

HeteroData(

author={

x=[4057, 334],

y=[4057],

train\_mask=[4057],

val\_mask=[4057],

test\_mask=[4057]

},

paper={ x=[14328, 4231] },

term={ x=[7723, 50] },

conference={

num\_nodes=20,

x=[20, 1]

},

(author, to, paper)={ edge\_index=[2, 19645] },

(paper, to, author)={ edge\_index=[2, 19645] },

(paper, to, term)={ edge\_index=[2, 85810] },

(paper, to, conference)={ edge\_index=[2, 14328] },

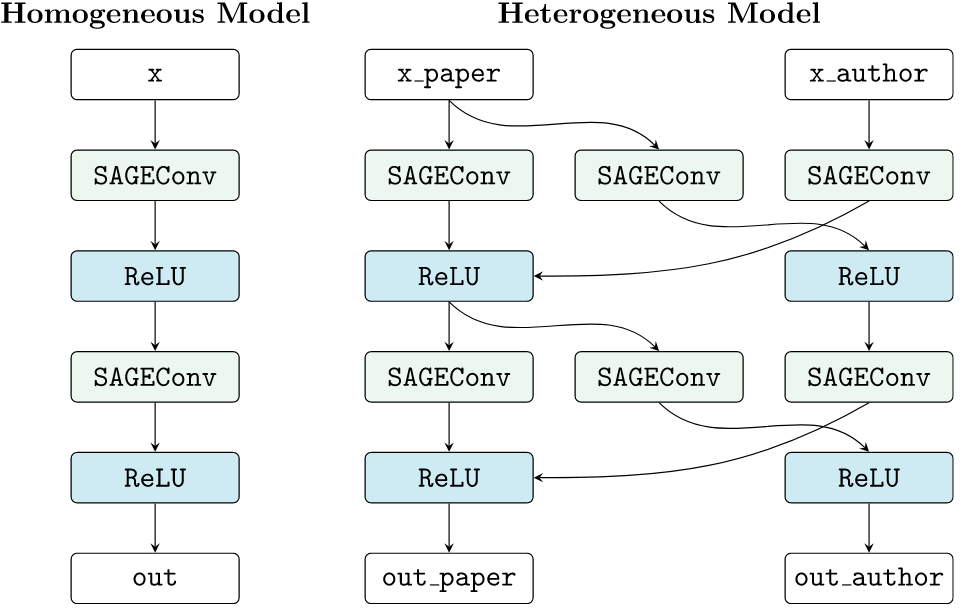
(term, to, paper)={ edge\_index=[2, 85810] },

(conference, to, paper)={ edge\_index=[2, 14328] }

)

## قسمت ب

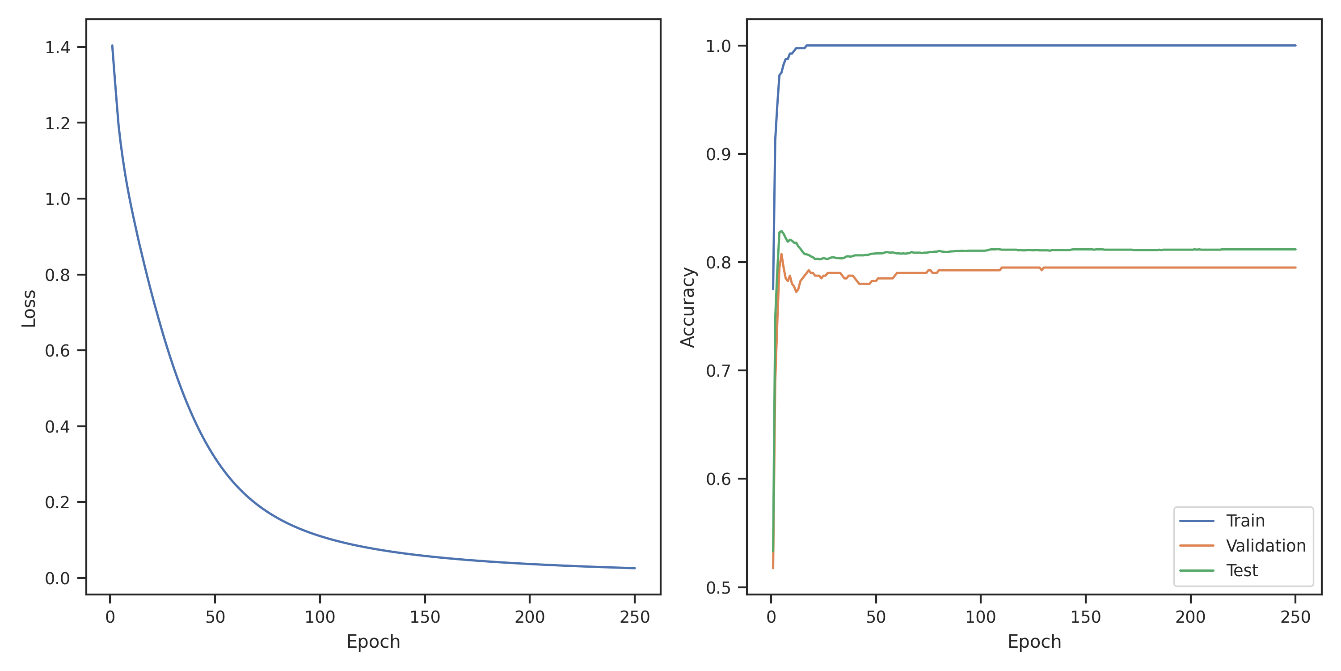
با توجه به اینکه با یک گراف ناهمگون سروکار داریم، لازم است پس از ایجاد هر شبکه عصبی گرافی همگون، آن را با استفاده از تابع to\_hetero که در کتابخانه PyG موجود است به یک مدل ناهمگون تبدیل کنیم. با این کار توابع پیام منحصربه‌فردی برای هر نوع یال ایجاد می‌شود. شکل زیر نمونه‌ای از یک GNN همگون و ناهمگون را در کنار یکدیگر نمایش می‌دهد.



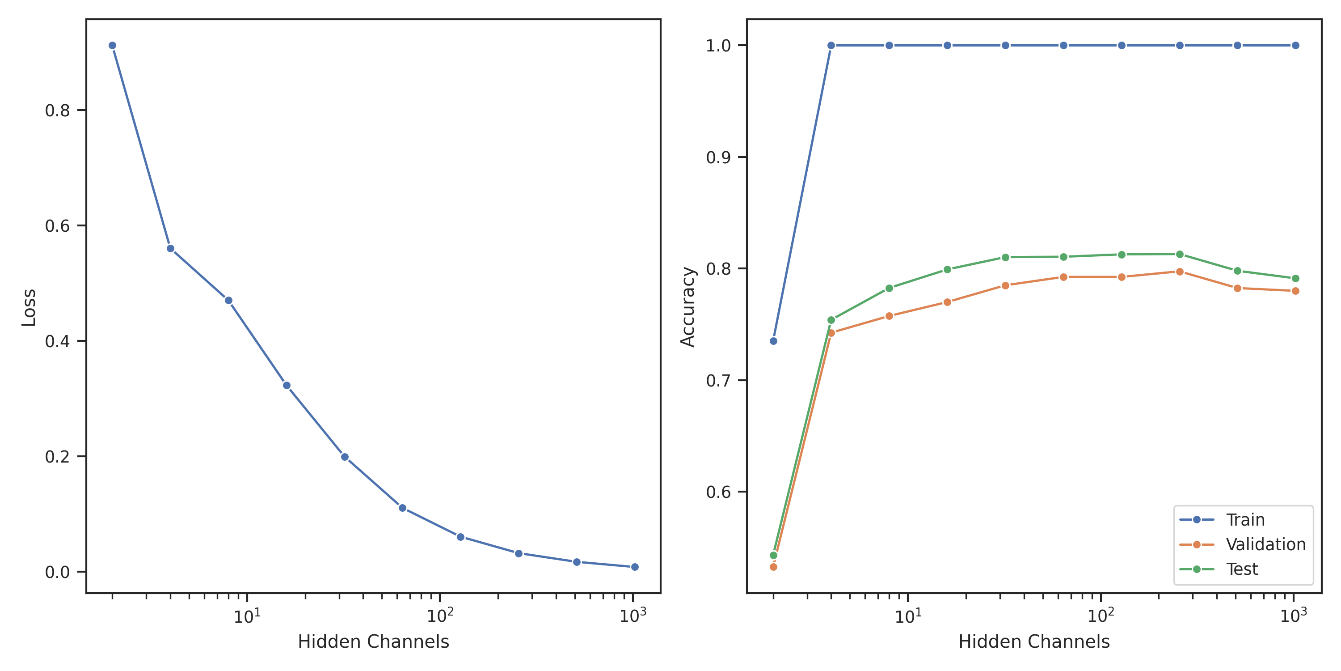
**شبکه‌های پیچشی گرافی (GCN)**

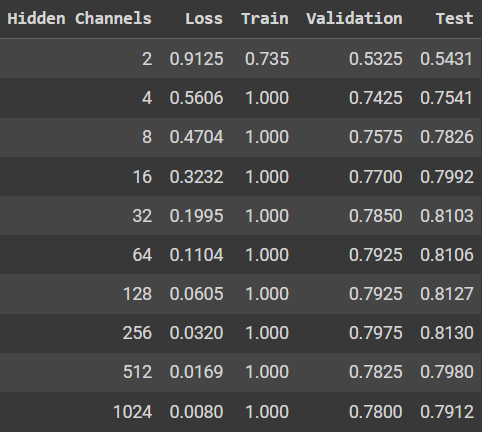
برای ایجاد یک شبکه پیچشی گرافی همگون از ماژول GCNConv استفاده می‌شود. اما این ماژول در کتابخانه PyG به گونه‌ای پیاده‌سازی شده است که امکان تبدیل به یک شبکه ناهمگون را ندارد. برای حل این مشکل، از گونه کلی‌تر یعنی SAGEConv استفاده می‌کنیم و اگر پارامتر تجمیع را با میانگین مقداردهی کنیم، مشابه GCN می‌شود.

ابتدا با استفاده از نمودار زیان برحسب دوره[[2]](#footnote-3) مقدار مناسب برای تعداد دوره‌ها را مشخص می‌کنیم. با توجه به نمودار زیر که با مقدار نرخ یادگیری 0.005 و دو لایه پیچشی ایجاد شده است، به‌‌نظر می‌رسد که مقدار 100 برای تعداد دوره‌ها مناسب باشد؛ زیرا پس از آن مقدار زیان، کاهش چندانی ندارد و همگرایی رخ داده است.

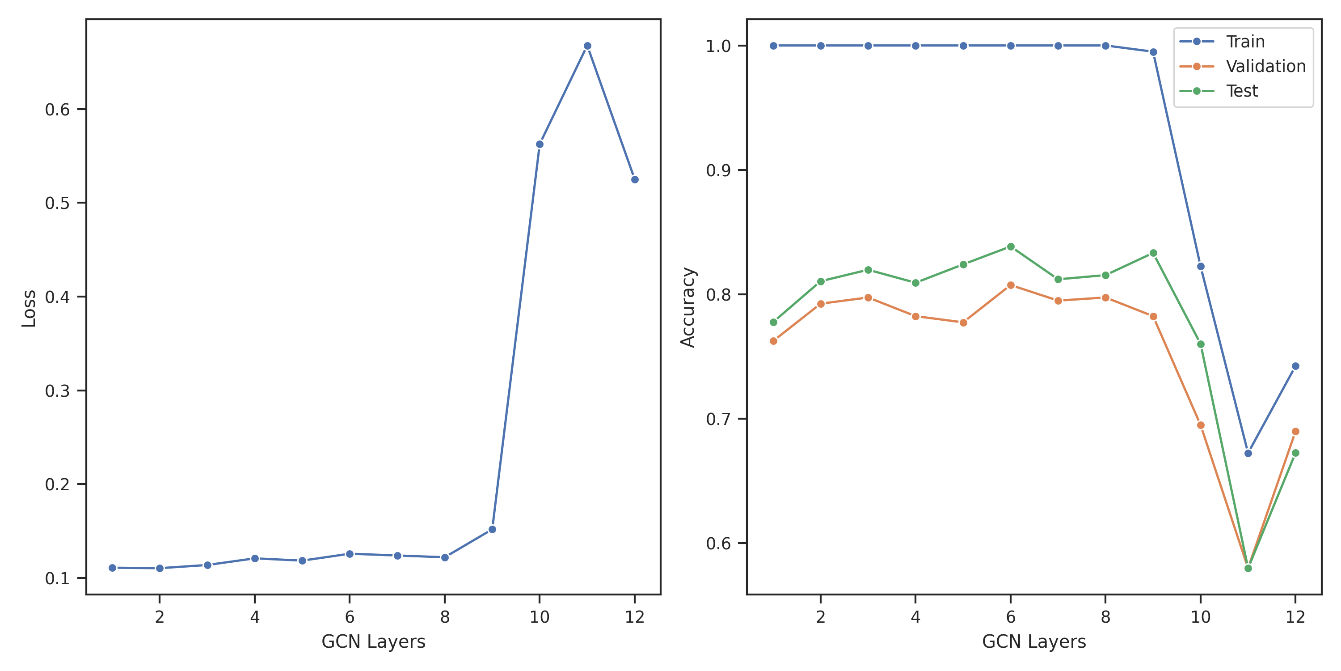


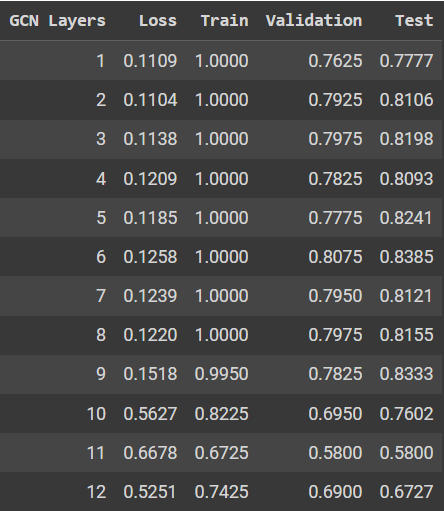
برای تشخیص مقدار مناسب برای تعداد ویژگی‌های مربوط به هر لایه پنهان، نمودار دقت بر حسب تعداد لایه‌های پنهان را رسم می‌کنیم. مقادیر مختلف برای ویژگی‌های لایه پنهان از 2 تا 1024 در مقیاس لگاریتمی مورد آزمایش قرار گرفته است. در اصل یادگیری ماشین، برای تنظیم هایپرپارامتر‌های یک مدل باید از مجموعه‌داده اعتبارسنجی[[3]](#footnote-4) استفاده کنیم. بنابراین با توجه به منحنی نارنجی یا مقادیر موجود در ستون اعتبارسنجی در جدول زیر، به‌ازای مقادیر 64 یا 256 بیشترین دقت را خواهیم داشت و با توجه به اینکه تفاوت دقت‌های حاصل از این دو مقدار بسیار ناچیز است، مقدار 64 را برای این هایپرپارامتر انتخاب می‌کنیم تا مدل ساده‌تری حاصل شود.



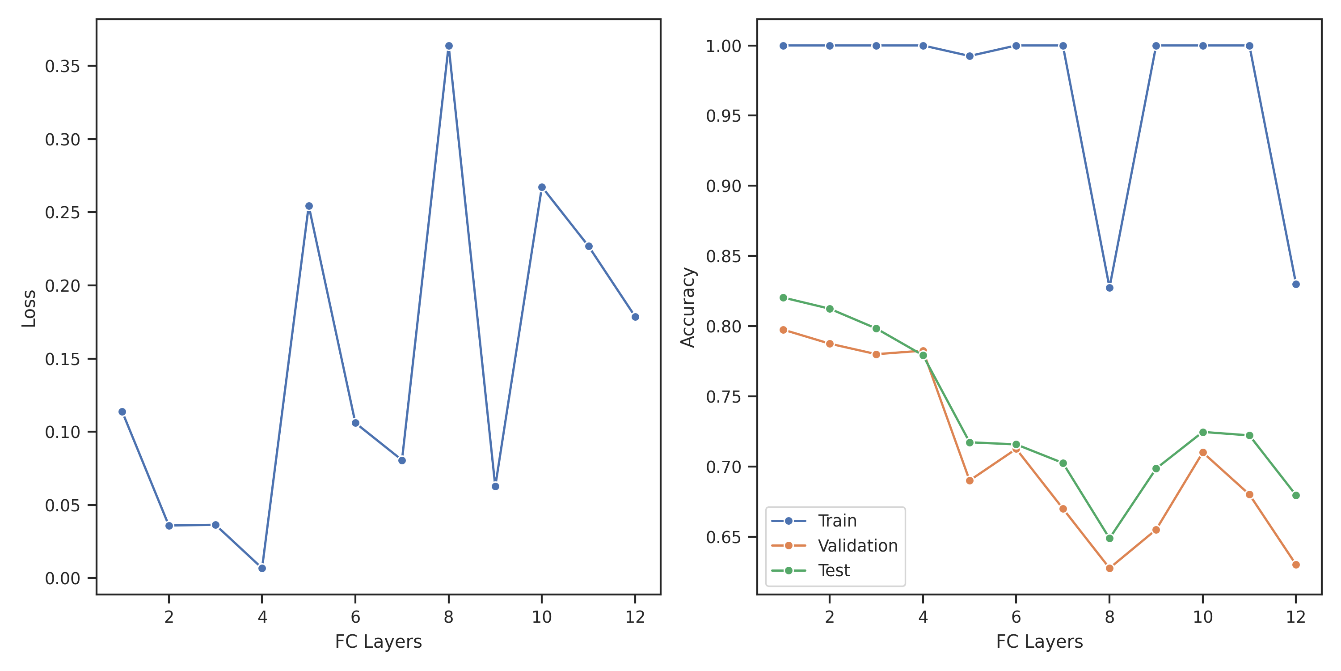


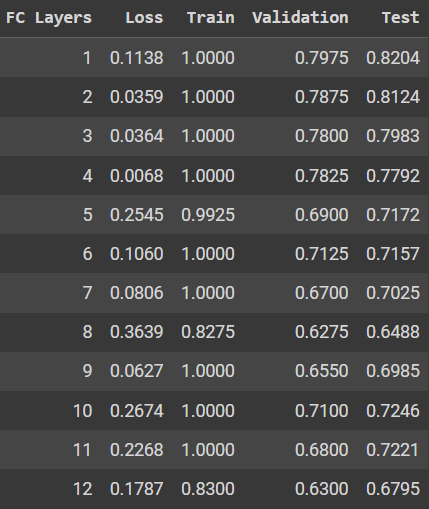
برای تشخیص تعداد مناسب لایه‌های پیچشی، مقادیر مختلف را آزمایش کرده و نمودار دقت را رسم می‌کنیم. با توجه به منحنی نارنجی و جدول مربوطه، هنگامی که 6 لایه پیچشی داشته باشیم، بیشترین دقت حاصل می‌شود. اما از آنجا که در صورت سؤال درخواست شده است که تنها مقادیر 1 تا 3 را درنظر بگیریم، مقدار 3 که در این بازه بیشترین دقت را دارد انتخاب می‌کنیم.





برای ایجاد لایه کاملاً متصل[[4]](#footnote-5)، از ماژول Linear استفاده می‌شود. با توجه به منحنی نارنجی در شکل زیر و جدول مربوطه، زمانی که تنها از یک لایه کاملاً متصل استفاده کنیم، بیشترین دقت حاصل می‌شود.





بنابراین GNN نهایی که متشکل از 3 لایه GCN با تعداد ویژگی‌های پنهان 64 و یک لایه کاملاً متصل است، دارای دقت 82 درصد بر روی مجموعه‌داده آزمون[[5]](#footnote-6) است.

GCN:

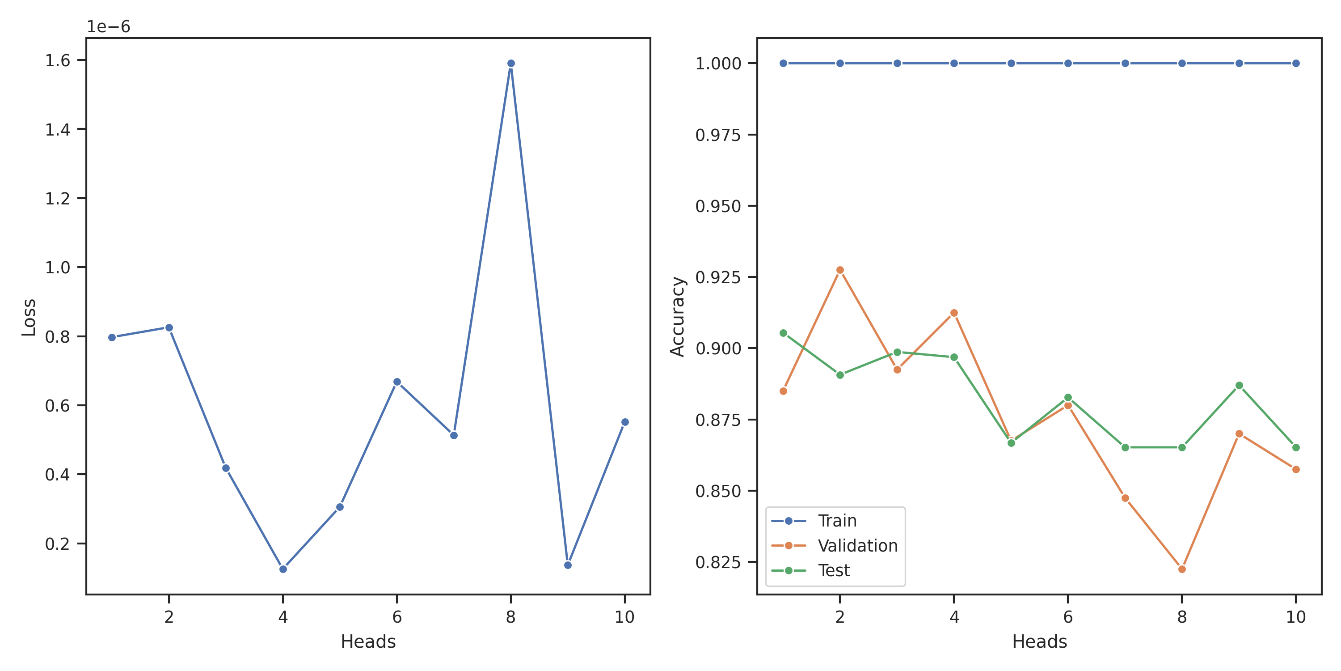
Accuracy: 0.8204

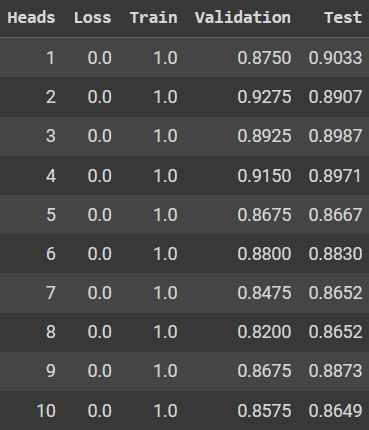
Computation Time: 14.5502s

با بررسی و تحلیل نمودارهای حاصل از مراحل قبل، می‌توان نتیجه گرفت که همیشه با افزایش یا کاهش مقادیر مربوط به هایپرپارامترها، دقت مدل افزایش یا کاهش پیدا نمی‌کند. برای هر هایپرپارامتر یک مقدار بهینه وجود دارد که با آزمون مقادیر مختلف یا گریدسرچ می‌توان آنها را پیدا کرد.

**شبکه‌های توجه گرافی (GAT)**

بررسی مقادیر مختلف برای تعداد سرهای مکانیزم توجه، نشان می‌دهد که زمانی که تعداد 2 سر داشته باشیم، بیشترین میزان دقت را بر روی مجموعه‌داده اعتبارسنجی خواهیم داشت.





درنهایت GNN حاصل که متشکل از 3 لایه GAT با تعداد ویژگی‌های پنهان 64 و تعداد سر 2 است، دارای دقت 89 درصد بر روی مجموعه‌داده آزمون است.

GAT:

Accuracy: 0.8907

Computation Time: 12.7384s

با توجه به اینکه زیرفضای متناسب با هر سر اهمیت متفاوتی دارد، می‌توان برای هر یک از آنها یک ضریب قابل آموزش درنظر گرفت و تجمیع آنها را به‌صورت وزن‌دار انجام داد. بدین منظور کافی است یک لایه Linear بعد از لایه GAT قرار داد؛ زیرا لایه GAT به‌ازای هر سر، کانال‌های خروجی متفاوتی ایجاد می‌کند و با اتصال آنها به لایه Linear وزن‌هایی متناسب با اهمیتشان یادگرفته شده و به آنها اختصاص داده می‌شود. دقت این GNN به 91 درصد رسیده است.

GAT using learnable coefficient:

Accuracy: 0.9116

Computation Time: 12.8774s

**شبکه‌های توجه گرافی 2 (GAT-V2)**

در نسخه دوم GAT ضرایب توجه برخلاف نسخه اول که لایه‌های خطی درست پس از یکدیگر اعمال می‌شدند، به‌صورت پویا محاسبه می‌شود.

GATv2 using Hadamard:

Accuracy: 0.9143

Computation Time: 15.5159s

GATv2 using Min:

Accuracy: 0.9183

Computation Time: 17.0286s

GATv2 using Max:

Accuracy: 0.9245

Computation Time: 17.0818s

## قسمت ج

در جدول زیر مدل‌های ساخته شده به‌ترتیب دقت بیان شده است. دقت شبکه‌های GAT به‌طور قابل ملاحظه‌ای از GCN بیشتر است؛ زیرا برخلاف GCN که اهمیت پیوندها تنها براساس ساختار گراف و بدون درنظر گرفتن ویژگی‌ گره‌ها تعیین می‌شود، در GAT به ویژگی گره‌ها نیز توجه می‌شود و برای هر گره لینک‌های ورودی به آن از اهمیت یکسانی برخوردار نیستند و لینک‌های با اهمیت بیشتر توجه بیشتری کسب می‌کنند و هنگام تجمیع پیام‌ها، وزن بیشتری به آنها اختصاص داده می‌شود. همان‌طور که مشاهده می‌شود، دقت شبکه‌های GAT ایجاد شده تفاوت چندانی با یکدیگر ندارد. عملکرد نسخه دوم شبکه‌های GAT به‌دلیل محاسبه ضرایب توجه به‌صورت پویا کمی بهتر است. در مجموع نتایج ارزیابی ما نشان می‌دهد که نسخه دوم شبکه GAT با استفاده از عملگر Max با دقت 92 درصد بهترین عملکرد را بر روی این مجموعه‌داده داشته است.

|  |  |
| --- | --- |
| **GNN** | **Accuracy** |
| GATv2 using Max | 0.9245 |
| GATv2 using Min | 0.9183 |
| GATv2 using Hadamard | 0.9143 |
| GAT using learnable coefficient | 0.9116 |
| GAT | 0.8907 |
| GCN | 0.8204 |

# سؤال دوم

**Simplifying Graph Convolutional Networks**

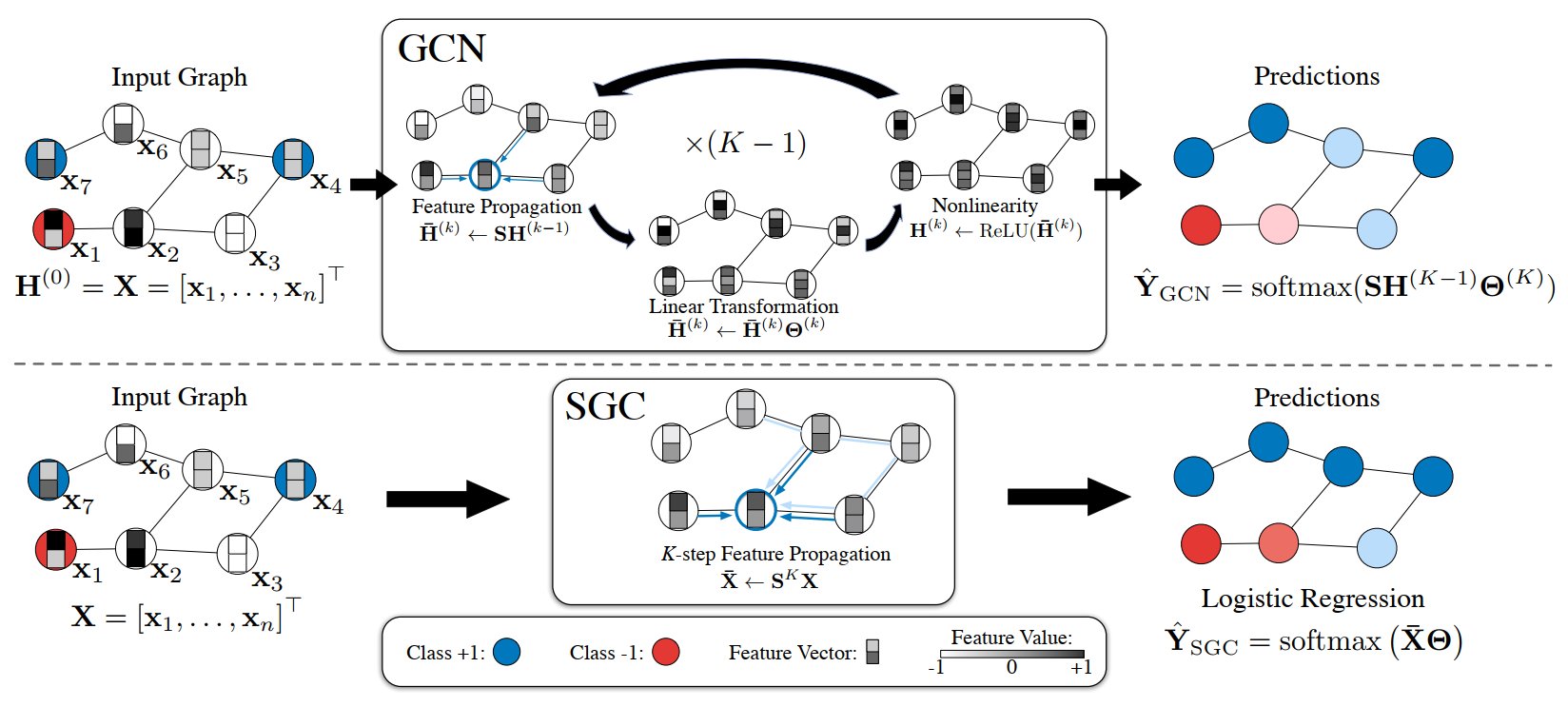
**DBLP**

## قسمت الف

شبکه‌های پیچشی گرافی (GCN) و گونه‌های مختلف آنها روش‌هایی هستند که برای یادگیری تعبیه یا بازنمایی گراف‌ها استفاده می‌شوند. این روش‌ها عملکرد خوبی داشته‌اند و از این جهت توجه زیادی را به خود جلب کرده‌اند. این روش‌ها الهام گرفته از رویکردهای اخیر یادگیری عمیق هستند و از آنجا که گراف‌ها ساختاری منعطف دارند، می‌توان آنها را گونه‌های تعمیم‌یافته درنظر گرفت. این شبکه‌ها از لایه‌های مختلفی تشکیل شده‌اند که بعضی از آنها ممکن است غیر ضروری باشند و بدون اینکه کارایی مدل را به‌طور قابل توجهی بهبود دهند تنها منجر به افزایش پیچیدگی محاسباتی شوند. در این پژوهش با حذف برخی لایه‌های غیرضروری در GCN و ساده‌سازی آن مدلی با نام SGC ارائه می‌شود که دقت آن نزدیک به مدل اصلی است و می‌تواند بسیار سریع‌تر عمل کند.

تفاوت این دو مدل در شکل زیر مشهود است. هر لایه از یک شبکه پیچشی K لایه‌ای طی سه مرحله تعبیه حاصل از لایه قبل را برای هر گره بروزرسانی می‌کند. در مرحله اول، انتشار ویژگی‌ها[[6]](#footnote-7) انجام می‌شود. یعنی در ابتدای هر لایه، ویژگی‌های هر گره با بردارهای ویژگی گره‌های همسایه‌های محلی آن با استفاده از میانگین، تجمیع می‌شود. تفاوت GCN و MLP نیز در وجود همین مرحله است. در مرحله دوم، تبدیل ویژگی‌ها[[7]](#footnote-8) انجام می‌شود. بردارهای تعبیه حاصل از مرحله قبل، با استفاده از ماتریسی از وزن‌ها که یادگرفته می‌شود، تبدیل می‌شوند (شبیه MLP). در مرحله سوم نیز بردارهای تعبیه از یک تابع فعال‌ساز غیرخطی عبور داده می‌شوند. پس از اینکه این سه مرحله در K لایه از GCN انجام شد، تعبیه نهایی هر گره مشخص می‌شود. در آخر با عبور این تعبیه‌ها از تابع SoftMax نرمال‌سازی انجام شده و طبقه هر گره مشخص می‌شود.

در شبکه SGC معادل با یک شبکه GCN متشکل از K لایه، تنها انتشار ویژگی‌ها انجام می‌شود و دیگر مانند قبل در هر لایه خبری از تبدیل ویژگی‌ها و عبور از تابع فعال‌ساز نیست. انتشار ویژگی‌ها را نیز می‌توان خیلی سریع با استفاده از ضرب ماتریسی انجام داد. بدین صورت که ماتریس مجاورت نرمال‌شده (S) به‌توان K می‌رسد و در ماتریس ویژگی‌ها (X) ضرب می‌شود. درنهایت ماتریس حاصل، از یک لایه خطی عبور داده می‌شود (در ماتریس ضرایب یاد گرفته شده ضرب می‌شود) و پس از عبور از تابع SoftMax طبقه هر گره مشص می‌شود.



برای مقایسه دقت و زمان اجرای GCN و SGC مجموعه‌داده DLBP انتخاب شده است. این مجموعه‌داده، زیرمجموعه‌ای از وب‌سایت کتابشناسی علوم کامپیوتر DBLP است که شامل یک گراف ناهمگون غیرجهت‌دار با چهار نوع موجودیت نویسندگان (4057 گره)، مقالات (14328 گره)، اصطلاحات (7723 گره) و کنفرانس‌ها (20 گره) می‌باشد. از این گراف در مسئله طبقه‌بندی گره‌ها استفاده می‌شود و هدف آن یادگیری یک مدل به‌منظور طبقه‌بندی نویسنده‌ها در چهار زمینه پایگاه‌داده، داده‌کاوی، هوش مصنوعی و بازیابی اطلاعات است. این گراف به‌صورت دوبخشی است؛ یعنی بین گره‌های هم‌نوع، هیچ یالی وجود ندارد. یال‌ها نیز فاقد ویژگی هستند. اطلاعات بیشتر درمورد این گراف در سؤال اول ذکر شده است و از بازگویی آنها خودداری می‌کنیم.

## قسمت ب

پیاده‌سازی SGC با ارثبری از کلاس MessagePassing از کتابخانه PyG انجام شده است. برای ایجاد شبکه پیچشی نیز از ماژول آماده SAGEConv می‌توان استفاده نمود (دلیل اینکه چرا از ماژول GCNConv استفاده نشده است، در سؤال اول توضیح داده شد). با مشاهده کدهای مقاله اصلی و برداشت خود از مقاله، این دو مدل را پیاده‌سازی کرده‌ایم. مطابق با دستورات آزمایش، تعداد لایه‌های GCN و تعداد بازنشرها در SGC برابر با 2 درنظر گرفته شده است (K=2). مقایسه این دو مدل باید از منظر دقت و زمان اجرا انجام شود. هرچه تعداد دوره‌ها برای یک مدل کمتر باشد، طبیعی است که زمان کمتری مصرف می‌کند. بنابراین تعداد دوره‌ها برای هر دو مدل را یکسان و برابر با 100 درنظر گرفتیم تا مقایسه عادلانه‌ای انجام شود. در عوض نرخ یادگیری مدل‌ها متفاوت است و آنها را با کمک گرفتن از نمودار زیان برحسب دوره (مانند سؤال اول) به‌گونه‌ای تنظیم کرده‌ایم که در 100 دوره به آستانه همگرایی برسند. برای اجرا از محیط گوگل کولب و GPU استفاده شده است. درنهایت نتایج ذیل حاصل گردید:

GCN:

Accuracy: 0.7817

Computation Time: 23.9660s

SGC:

Accuracy: 0.7657

Computation Time: 15.4944s

همان‌طور که ملاحظه می‌شود، تفاوت دقت SGC و GCN کمتر از 2 درصد است. در عوض SGC تقریباً 1.5 برابر سریع‌تر عمل کرده است. با اینکه در مقاله اصلی بر روی مجموعه‌داده DBLP آزمایشی انجام نشده است، با بررسی مشابهت‌ها می‌توان نتیجه گرفت که نتایج آزمایش ما با نتایج مقاله اصلی تقریباً همخوانی دارد و قابل قبول است.

## قسمت ج

ملاحظه شد که در SGC برخی از پیچیدگی‌های غیرضروری GCN کاهش یافت و این امر منجر به کاهش زمان اجرا گردید بدون اینکه دقت تغییر چندانی کند. SGC طی K مرحله، فرآیند انتشار ویژگی‌ها را با درنظر گرفتن همسایه‌های محلی هر گره انجام می‌دهد. در برخی کاربردها یا مجموعه‌داده‌ها، ممکن است انتشار ویژگی‌ها و درنظر گرفتن همسایه‌های محلی، تأثیر چندانی بر روی دقت مدل نداشته باشد و صرفاً با درنظر گرفتن ویژگی گره‌ها بتوان یک مدل یادگیری پرسرعت با دقت مناسب به‌دست آورد. برهمین اساس شبکه‌ای با استفاده از دو لایه خطی ایجاد کردیم (شبیه MLP) و با اجرای آن بر روی مجموعه‌داده DBLP نتایج زیر حاصل گردید:

Our GNN:

Accuracy: 0.7639

Computation Time: 0.8900s

همان‌طور که ملاحظه می‌شود، دقت این شبکه تقریباً با SGC برابر است اما زمان اجرای آن بسیار کمتر است. در این مورد تقریباً 15 برابر سریع‌تر عمل کرده است. البته لازم به‌ذکر است که این نتایج تنها مربوط به DBLP است و مجموعه‌داده‌های دیگر ممکن است نتایج متفاوتی داشته باشند.

بنابراین نتیجه نهایی که می‌توان از این آزمایش‌ها گرفت این است که برای هر کاربرد یا مجموعه‌داده، بهتر است قبل از اینکه مستقیم به سراغ مدل‌های پیچیده برویم، مدل‌های ساده‌تر را نیز امتحان کنیم. شاید خیلی از این پیچیدگی‌ها غیرضروری باشند و دقت را به‌طور قابل ملاحظه‌ای افزایش ندهند و صرفاً منجر به افزایش زمان اجرا شوند.

1. Heterogeneous [↑](#footnote-ref-2)
2. Epoch [↑](#footnote-ref-3)
3. Validation [↑](#footnote-ref-4)
4. Fully-connected layer [↑](#footnote-ref-5)
5. Test [↑](#footnote-ref-6)
6. Feature propagation [↑](#footnote-ref-7)
7. Feature transformation [↑](#footnote-ref-8)