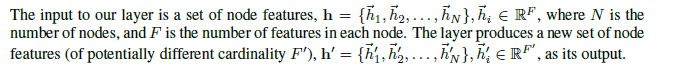
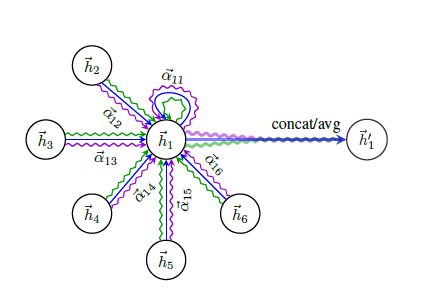


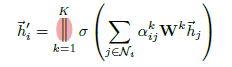
num\_in\_features: تعداد ویژگی‌های ورودی

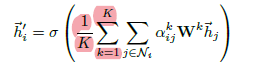
num\_out\_features: تعداد ویژگی‌های خروجی



num\_of\_heads: تعداد هدها که برای stable یا پایدار کردن فرایند یادگیری از چند هد ممکن است استفاده کنیم. در شکل سه هد داریم که با سه رنگ بنفش، آبی و سبز نشان داده شده‌اند.

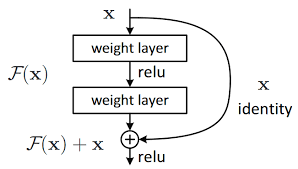
concat: این متغیر این را مشخص می‌کند که خروجی هدها را با هم concat کنیم یا از آن‌ها میانگین بگیریم



OR 

activation: تابع فعالسازی که در این‌جا از تابع elu (exponential relu) استفاده شده است.

dropout\_prob: نرخ حذف تصادفی

add\_skip\_connection: استفاده از تکنیک connection skip :

bias: اضافه کردن bias که استفاده از آن در GAT ضروری نیست.

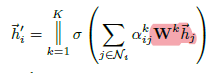
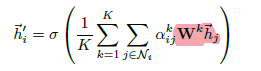
log\_attention\_weights: تعیین کننده اینکه از وزن‌های توجه لگاریتم بگیریم.

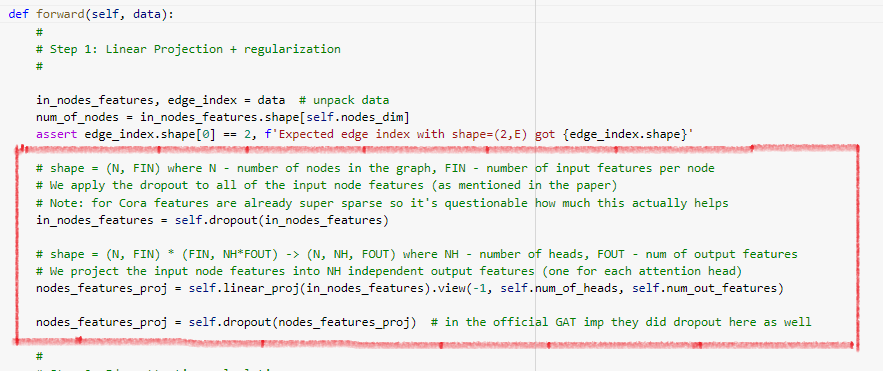
**گام اول: تبدیل خطی و regularization**



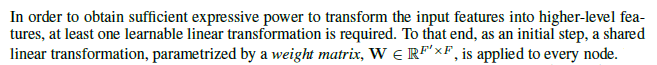
برای به دست آوردن ویژگی‌های high levelتر از روی ویژگی‌های ورودی روی آن‌ها حداقل یک تبدیل خطی اعمال می‌کنیم.

به تعداد هدها ماتریس وزن مستقل اعمال می‌شود یعنی به تعداد num\_of\_heads ماتریس وزن داریم.

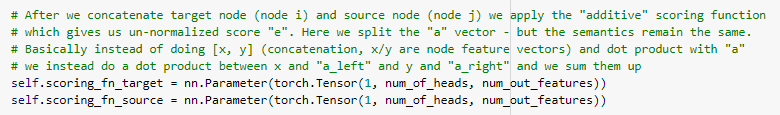
 

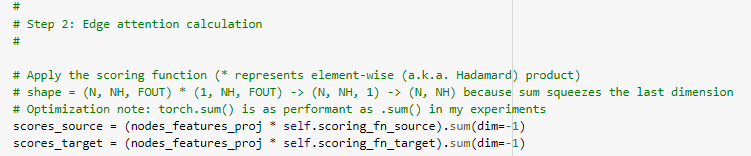


در قسمت فوق تبدیل خطی اولیه که در بالا به آن اشاره شد اعمال شده و قبل و بعد از آن روی گره‌ها dropout اعمال شده که در مقاله به dropout اشاره نشده است اما در کد اصلی مقاله هر دو dropout اشاره شده اعمال شده اند.

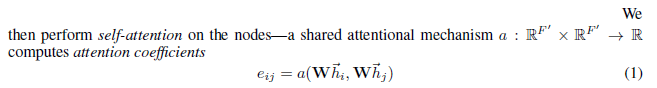


**گام دوم: محاسبه‌ی توجه**

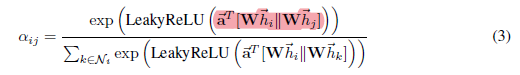




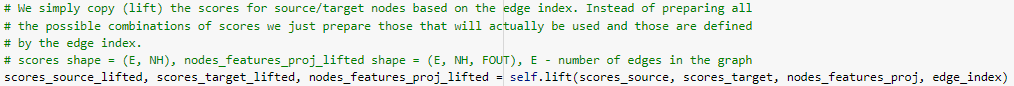
در این قست به محاسبه ضرایب توجه نرمال نشده یعنی eij در مقاله می‌پردازیم.



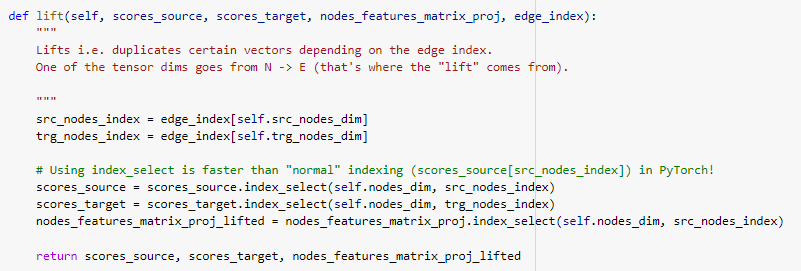
مقدار eij  برابر قسمت هایلایت شده در فرمول زیر است:



در کد پیاده‌سازی شده به جای اینکه بردار a را در الحاق دو بردار Whi و Whj ضرب داخلی کنیم، a را به دونیمه‌ی چپ و راست تقسیم می‌کنیم و سپس نیمه‌ی چپ را در Whi و نیمه راست را در Whj ضرب داخلی می‌کنیم و سپس نتایج حاصل را با هم جمع می‌کنیم.



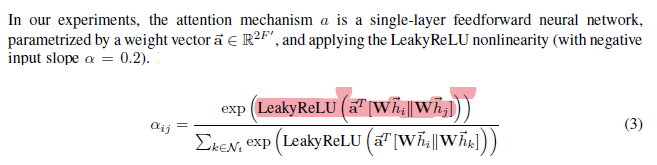
تابع lift که در کد فوق استفاده شده و پیاده‌سازی آن به شرح زیر است برای محاسبه‌ی ضرب‌های داخلی (a\_left . Whi) و (a\_right . Whj) با توجه به یالهای بین i و j است.

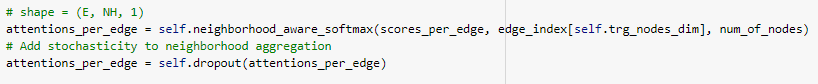




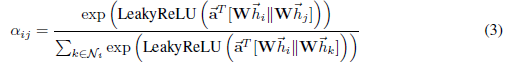


سپس روی حاصل جمع دو عبارت (a\_left . Whi) و (a\_right . Whj) تابع leakyReLU اعمال شده است.

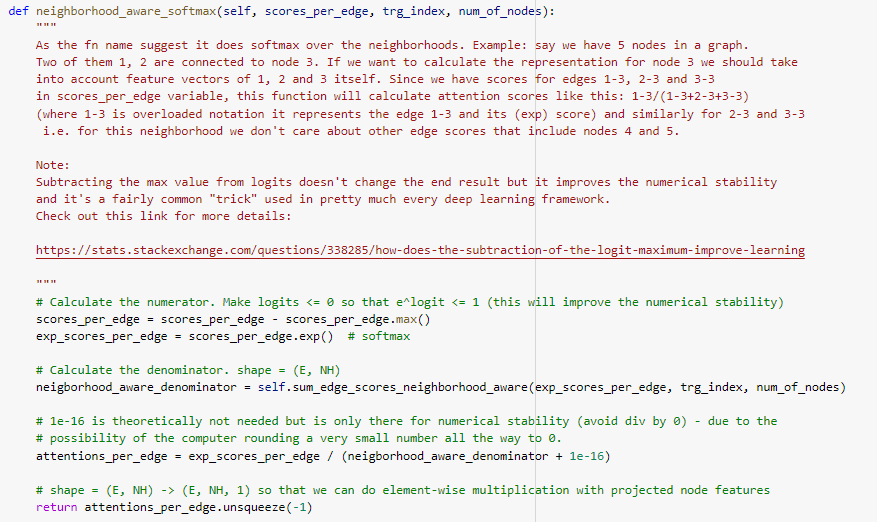




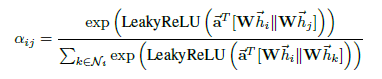
در نهایت با اعمال تابع softmax روی هر گره با توجه به همسایگانش ( که خود گره را نیز شامل می‌شود) ضرایب توجه نرمال شده را به دست می‌آوریم:



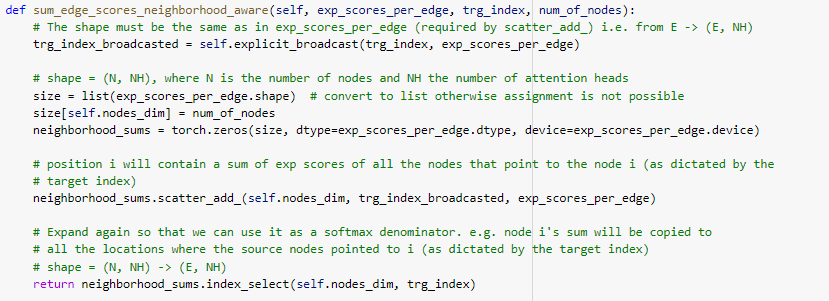
پس از به دست آوردن softmax روی خروجی آن که در گام بعدی استفاده می‌شود dropout اعمال شده است.



تابع فوق مربوط به اعمال softmax روی ضرائب توجه و به دست آوردن ضرائب توجه نرمال شده است.

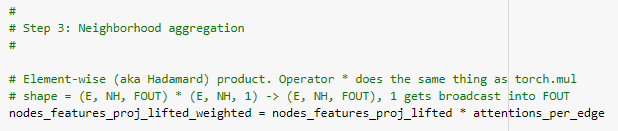


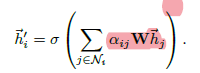
مخرج رابطه‌ی softmax فوق با استفاده از تابع زیر به دست آمده‌است.

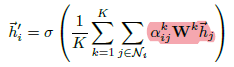


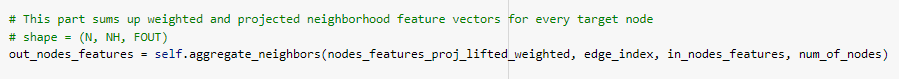
**گام سوم: انباشتن همسایه‌ها**

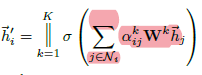
بعد از به دست آوردن ضرایب توجه نرمال شده جهت محاسبه‌ی بردار ویژگی جدید هر گره یا همان بردار ویژگی high levelتر این ضرایب را در بردار ویژگی تبدیل‌یافته ضرب می‌کنیم.

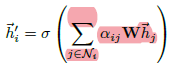
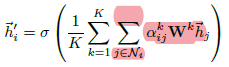


این محاسبات در فرمول‌های زیر مشخص شده‌اند:

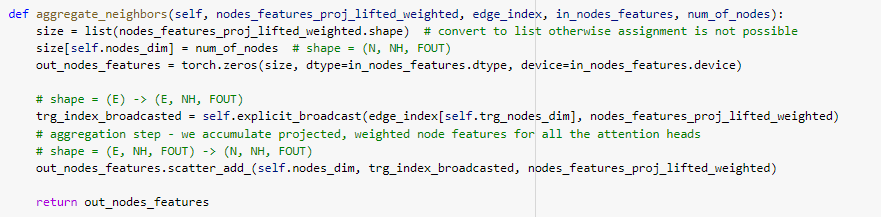




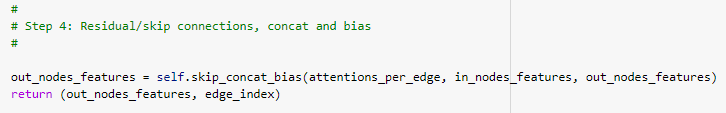
در آخرین قسمت از این مرحله برای به دست آوردن بردار ویژگی جدید هر گره، بردار ویژگی تبدیل‌ یافته‌ی وزن‌دار همسایگان آن را با هم جمع می‌کنیم که توسط تابع aggregate\_neighbors انجام شده است.

پیاده‌سازی تابع aggregate\_neighbors به صورت زیر است:



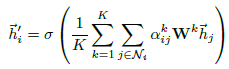
**گام چهارم: skip\_connection, concat, bias**

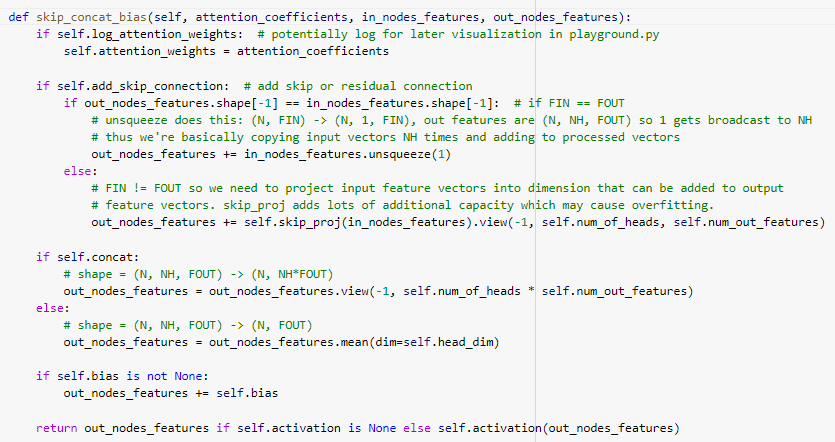


در این مرحله با توجه به متغیرها درباره سه مورد زیر تصمیم می‌گیریم:

* آیا از تکینک skip connection استفاده کنیم؟
* خروجی هدها را با هم الحاق کنیم یا از آن‌ها میانگین بگیریم؟
* استفاده از bias در لایه‌ی Gat

خروجی نهایی خروجی این مرحله است.



استفاده یا عدم استفاده از log\_attention\_weights مربوط به بصری سازی است که در این پیاده‌سازی توجه چندانی به آن نشده است.

استفاده یا عدم استفاده از skip connection، bias و همچنین استفاده از concat برای خروجی هدها یا میانگیری از آنها همگی در ابتدا توضیح داده شده‌اند فقط پیاده‌سازی آن‌ها در تابع فوق انجام شده است.