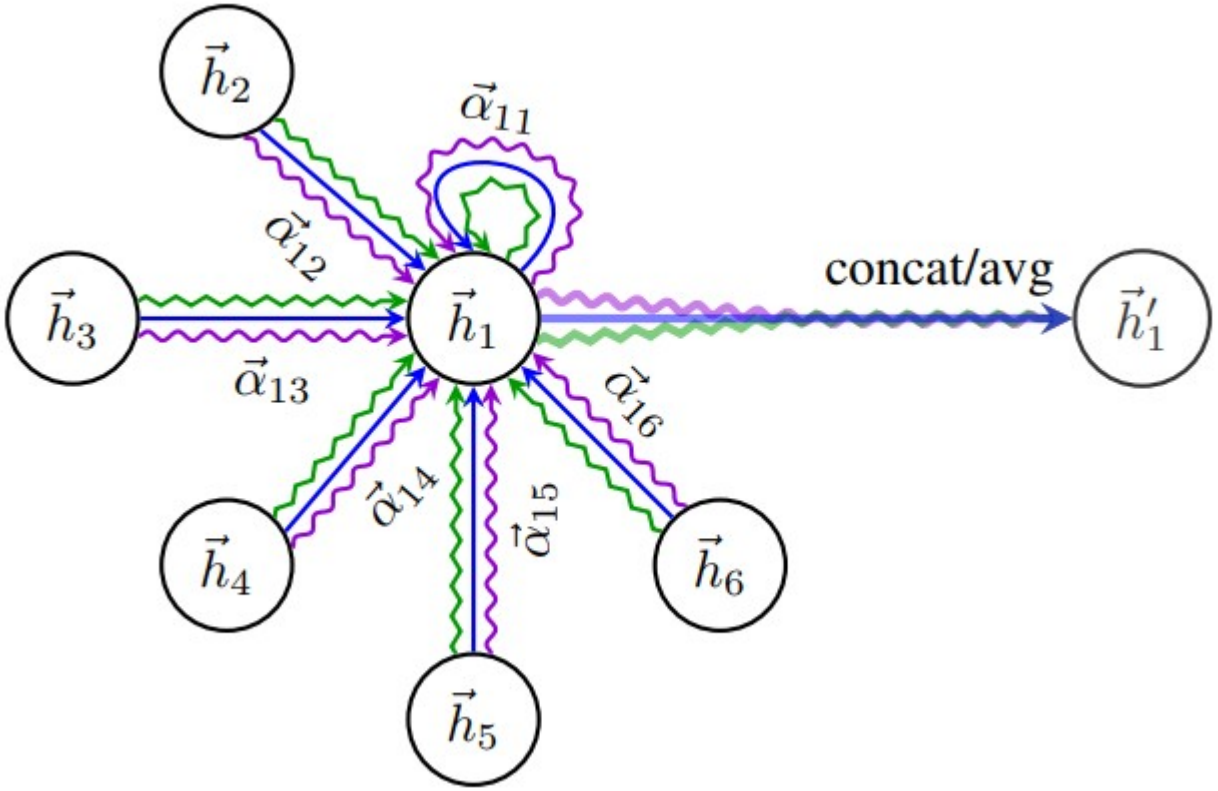
IIC-3641 Aprendizaje Automático basado en Grafos

https://github.com/marcelomendoza/IIC3641

- RECAPITULACIÓN -

Graph Attention Networks

En la AF6 hicimos clasificación de nodos usando las redes GAT sobre CiteSeer. El modelo usa Multihead Attention y evaluamos ele efecto de la cantidad de heads en la tarea:



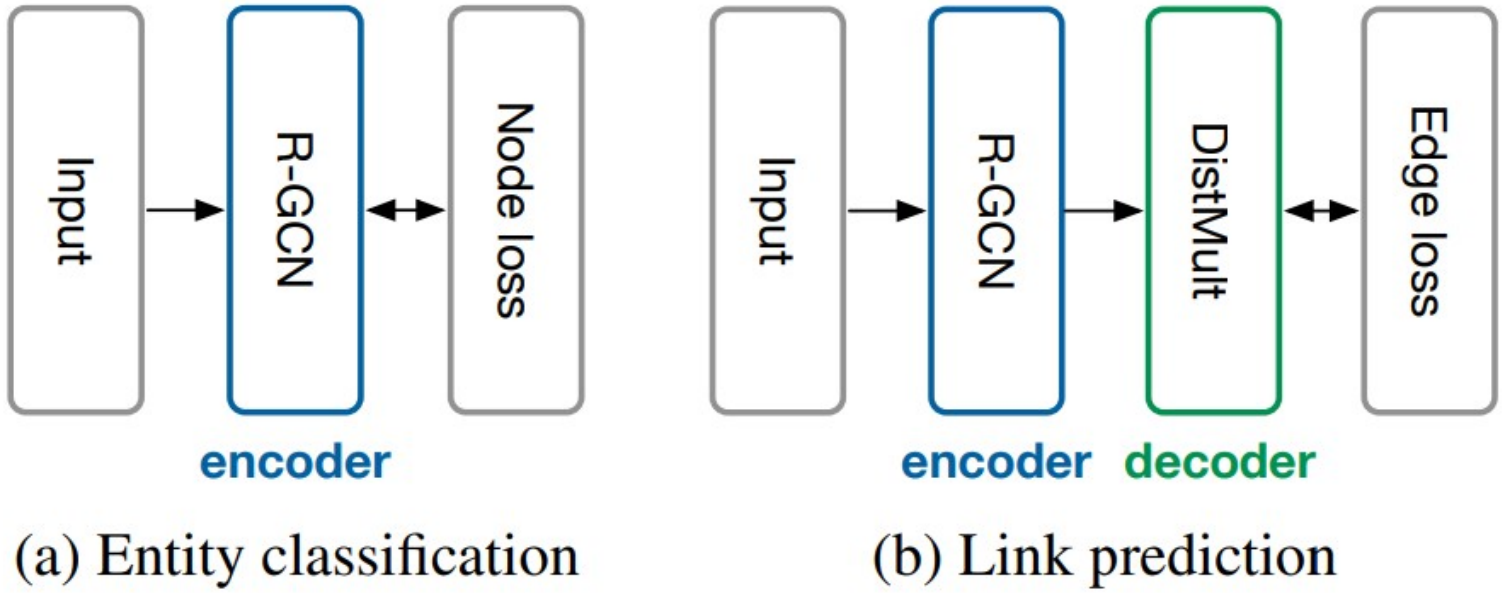
Concluimos que no necesariamente más heads implican mejos desempeño.
La solución daba algo entre 4 y 9 cabezales.

Relational GCN

En la AF6 también entrenamos una R-GCN. Nos dimos cuenta que el costo computacional era un tema, y que los datasets pueden ocupar muchos recursos computacionales.

En específico, usamos la R-GCN sobre un grafo multi-relacional. Algo interesante que vimos es que el grafo en realidad se homogeniza, es decir, no se trabaja de forma nativa como heterogéneo. Los atributos de los enlaces permiten representar distintos tipos de relaciones, lo cual aparece como un parámetro del modelo.

La R-GCN puede verse como un encoder, y ser usada a nivel de nodos o enlaces.



👉 Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. European Semantic Web Conference (pp. 593-607).

- GRAPH TRANSFORMER -

Graph Transformer

El graph transformer es una extensión del transformer original, diseñado para NLP, el cual incluye cuatro variaciones que lo hacen aplicable a grafos:

1. El mecanismo de atención es una función de la conectividad del vecindario de cada nodo.
2. Los positional encodings se representan en base a los vectores propios del Laplaciano del grafo.
3. La capa de normalización se reemplaza por una capa batch normalization, lo que premite que el entrenamiento sea más eficiente.
4. La arquitectura se extiende para manejar explícitamente representación de enlaces, a diferencia de las R-GCN que no operan de forma nativa con enlaces.

Un argumento interesante en la formulación del graph transformer está en que al usar el mecanismo de atención a nivel de vecindario, se hace un mejor uso del **sparseness**, lo que representa un **bias inductivo** útil para aprendizaje en datasets de grafos. La información global quedará representada a nivel de los positional encodings.

Dwivedi, V., Bresson, X. A generalization of Transformer Networks to Graphs, 2021
https://arxiv.org/pdf/2012.09699 .

Graph Transformer

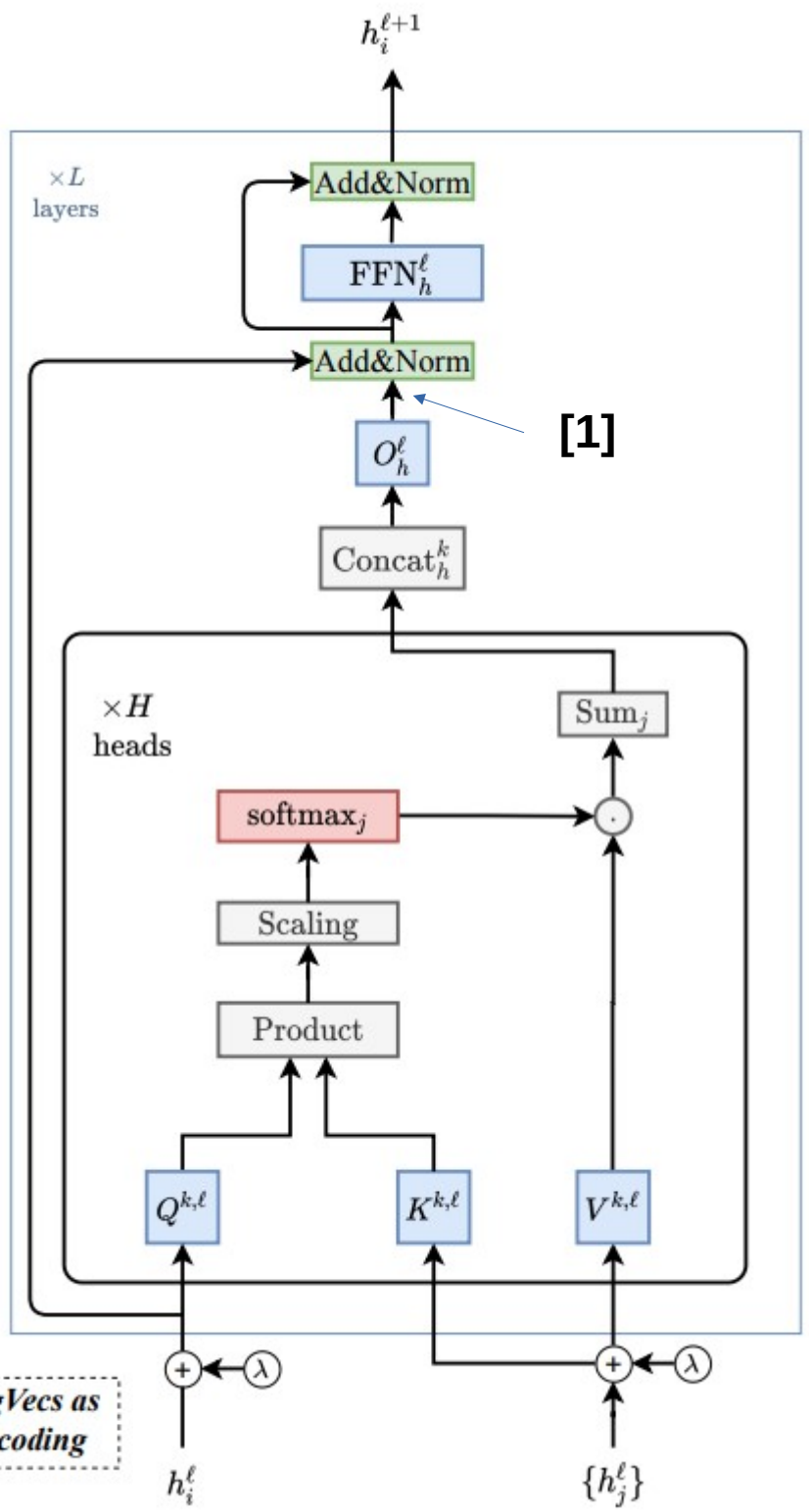
El graph transformer se propone sobre la base de dos variaciones una que sólo opera a nivel de representaciones de nodos, y otra que incluye representaciones para los enlaces.

Veamos primero el graph transformer para nodos. Observamos que esto es parecido al truco que se usa en T5. En este caso, la cross-attention se usa para relacionar el nodo *i* con el vecindario:

[1] $$\hat{h}_i^{\ell+1} = O_h^{\ell} \Big\Vert_{k=1}^{H} \Big(\sum_{j \in \mathcal{N}_i} w_{ij}^{k,\ell} V^{k,\ell} h_j^{\ell} \Big),$$

donde,

$$w_{ij}^{k,\ell} = \text{softmax}_j \Big(\frac{Q^{k,\ell} h_i^{\ell} \cdot K^{k,\ell} h_j^{\ell}}{\sqrt{d_k}} \Big)$$



[1]

Graph Transformer

Veamos la inicialización y el dimensionamiento de la red.

Inicialización: en base a datos del grafo.

Notación:

Características de los nodos: $\alpha_i \in \mathbb{R}^{d_n \times 1}$

Inicialización a nivel de nodos: $\hat{h}_i^0 = A^0 \alpha_i + a^0$

parámetros

$A^0 \in \mathbb{R}^{d \times d_n}$ $a^0 \in \mathbb{R}^d$ ← dim de capa lineal

Laplacian positional encodings: $\lambda_i^0 = C^0 \lambda_i + c^0$ $c^0 \in \mathbb{R}^d$

$C^0 \in \mathbb{R}^{d \times k}$

Vector propio de L precomputado de dimensionalidad k

La dimensionalidad *d* se va a mantener en la capa de atención, pero los vectores V, Q y K tendrán una dimensionalidad más baja, denominada dimensionalidad del cabezal (d_k):

$$Q^{k,\ell}, K^{k,\ell}, V^{k,\ell} \in \mathbb{R}^{d_k \times d}$$

Graph Transformer

El graph transformer también usa conexiones residuales:

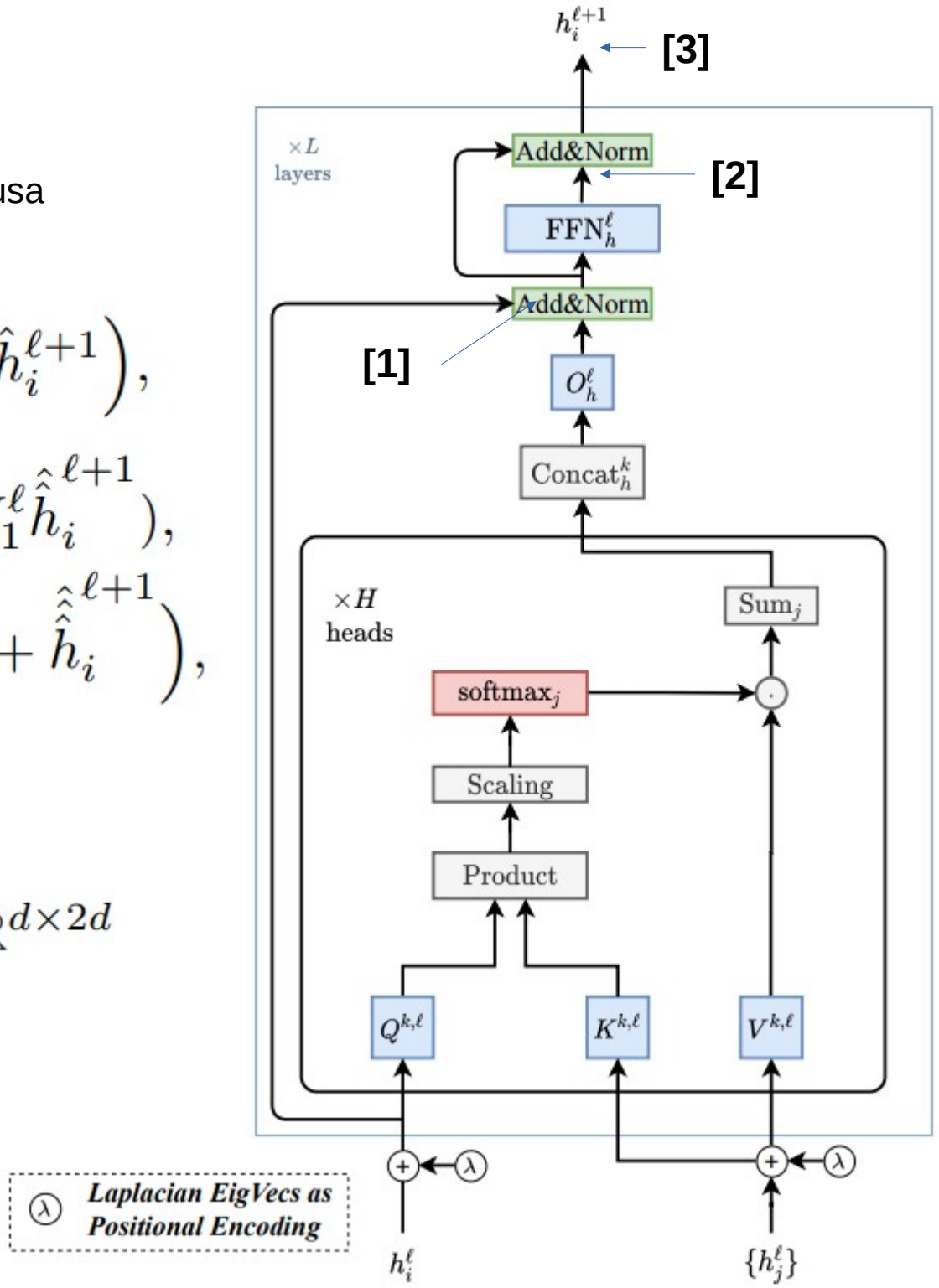
[1] $$\hat{\hat{h}}_i^{\ell+1} = \text{Norm}\left(h_i^{\ell} + \hat{h}_i^{\ell+1}\right),$$

[2] $$\hat{\hat{h}}_i^{\ell+1} = W_2^{\ell}\text{ReLU}(W_1^{\ell}\hat{\hat{h}}_i^{\ell+1}),$$

[3] $$h_i^{\ell+1} = \text{Norm}\left(\hat{\hat{h}}_i^{\ell+1} + \hat{\hat{h}}_i^{\ell+1}\right),$$

donde,

$$W_1^{\ell}, \in \mathbb{R}^{2d\times d}, W_2^{\ell}, \in \mathbb{R}^{d\times 2d}$$



Graph Transformer

Revisemos ahora el graph transformer que incluye representaciones para los enlaces. La parte de los nodos es parecida:

$$\hat{h}_i^{\ell+1} = O_h^{\ell} \Big\Vert_{k=1}^{H} \Big(\sum_{j \in \mathcal{N}_i} w_{ij}^{k,\ell} V^{k,\ell} h_j^{\ell} \Big),$$

[1]

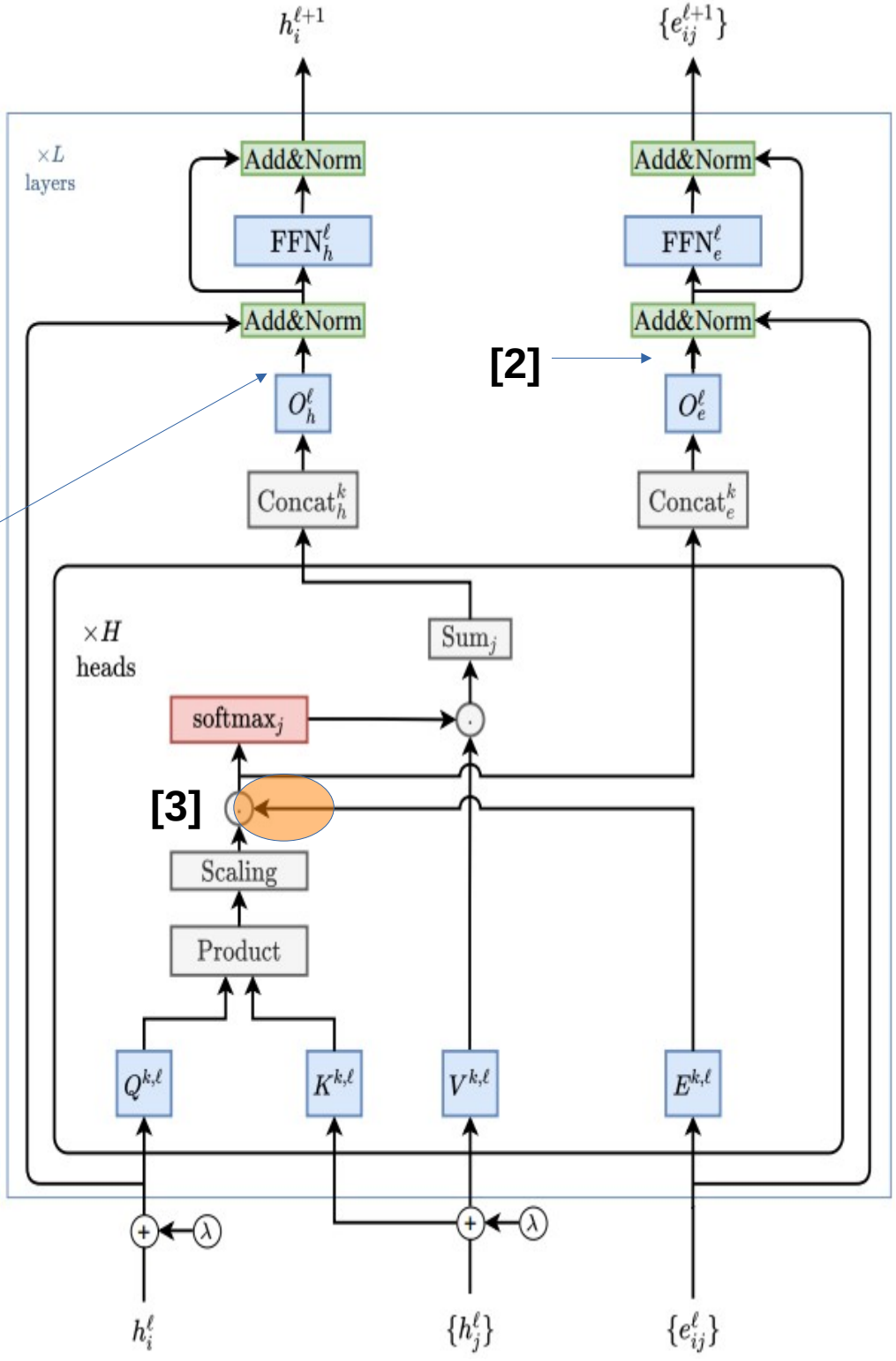
Pero tenemos algo nuevo para trabajar con los enlaces:

[2] $$\hat{e}_{ij}^{\ell+1} = O_e^{\ell} \Big\Vert_{k=1}^{H} \Big(\hat{w}_{ij}^{k,\ell} \Big),$$

Es decir, los enlaces tocan como se calculan los coeficientes de atención:

$$\hat{w}_{ij}^{k,\ell} = \Big(\frac{Q^{k,\ell} h_i^{\ell} \cdot K^{k,\ell} h_j^{\ell}}{\sqrt{d_k}} \Big) \cdot E^{k,\ell} e_{ij}^{\ell}$$

[3]



Graph Transformer

Veamos en más detalle el cálculo de los coeficientes de atención:

$$\hat{w}_{ij}^{k,\ell} = \left(\frac{Q^{k,\ell} h_i^{\ell} \cdot K^{k,\ell} h_j^{\ell}}{\sqrt{d_k}} \right) \cdot E^{k,\ell} e_{ij}^{\ell}$$

donde, $Q^{k,\ell}, K^{k,\ell}, V^{k,\ell}, E^{k,\ell} \in \mathbb{R}^{d_k \times d}$. Vemos que el graph transformer agrega una matriz de parametros más al mecanismo de atencion para poder trabajar con enlaces.

El resto de la red se maneja, igual, con conexiones residuales, pero ahora a nivel de nodos y enlaces:

$$\hat{\hat{h}}_i^{\ell+1} = \text{Norm}\left(h_i^{\ell} + \hat{h}_i^{\ell+1} \right), \quad \hat{\hat{e}}_{ij}^{\ell+1} = \text{Norm}\left(e_{ij}^{\ell} + \hat{e}_{ij}^{\ell+1} \right),$$

$$\hat{\hat{h}}_i^{\ell+1} = W_{h,2}^{\ell} \text{ReLU}(W_{h,1}^{\ell} \hat{\hat{h}}_i^{\ell+1}), \quad \hat{\hat{e}}_{ij}^{\ell+1} = W_{e,2}^{\ell} \text{ReLU}(W_{e,1}^{\ell} \hat{\hat{e}}_{ij}^{\ell+1}),$$

$$h_i^{\ell+1} = \text{Norm}\left(\hat{\hat{h}}_i^{\ell+1} + \hat{\hat{\hat{h}}}_i^{\ell+1} \right), \quad e_{ij}^{\ell+1} = \text{Norm}\left(\hat{\hat{e}}_{ij}^{\ell+1} + \hat{\hat{\hat{e}}}_{ij}^{\ell+1} \right),$$

Graph Transformer

El graph transformer está formulado para operar como encoder, es decir, no está entrenado de manera autoregresiva para generar por ejemplo grafos de forma generativa.

Al usar el graph transformer encoder, podemos extraer sus embeddings para downstream tasks. En general, es mejor como encoder que las GCN y las GAT.

El paper muestra al Graph Transformer en dos tareas: regresión de grafos y clasificación de nodos. Regresión de grafos es una tarea que consiste en usar un scorer a nivel de grafos. Esto es parecido a lo qu hacíamos con link prediction usando TransE o DistMult. Lo que hacen en esta tarea es hacer algo que se denomina **graph readout**, esto es, hacer agregación, en este caso a nivel de nodos, para hacer scoring de grafos sobre las *C* clases del problema. El graph readout es:

$$y_{\mathcal{G}} = \frac{1}{\mathcal{V}} \sum_{i=0}^{\mathcal{V}} h_i^L$$

Esto nos da un embedding de un grafo y nos permite hacer regresión de grafos:

$$y_{\text{pred}} = P\,\text{ReLU}\left(Q\,y_{\mathcal{G}}\right)$$

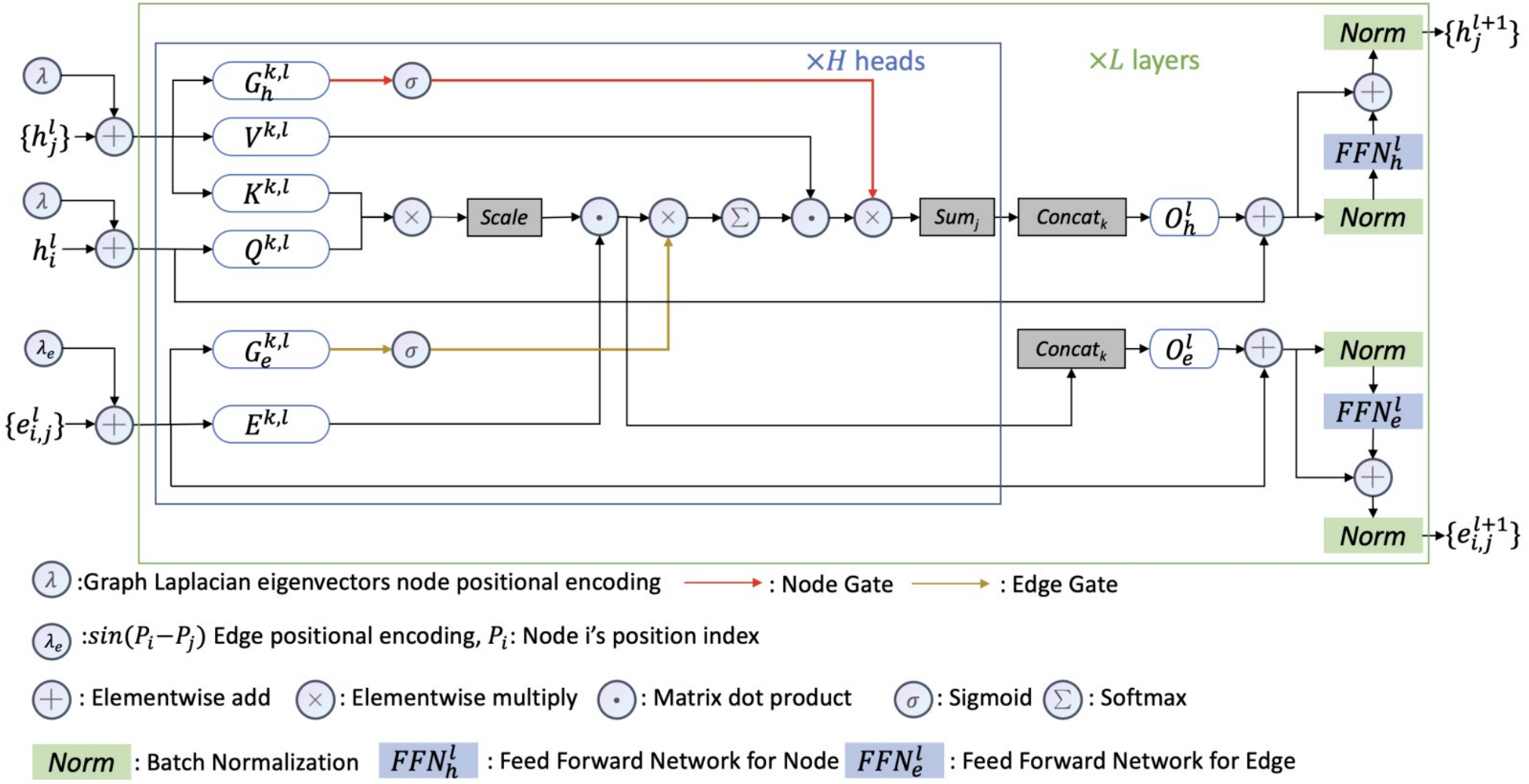
donde, $P \in \mathbb{R}^{d \times C}, Q \in \mathbb{R}^{d \times d}$

Ver experimentos en la referencia:

Dwivedi, V., Bresson, X. A generalization of Transformer Networks to Graphs, 2021
https://arxiv.org/pdf/2012.09699 .

Gated Graph Transformer

El gated graph transformer es una extensión del GT que incluye parámetros G los cuales se usan a la salida de los cálculos de coeficientes de atención.



- APLICACIÓN EL DESCUBRIMIENTO CIENTÍFICO -

Graph Transformer en ciencia

Article | Published: 20 December 2023

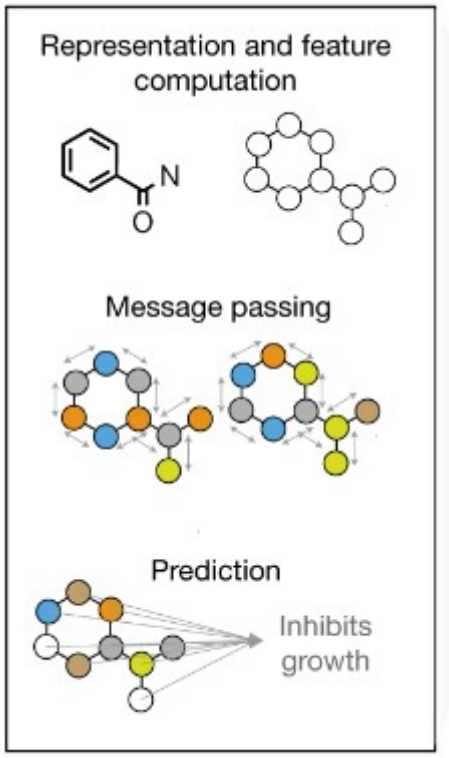
Discovery of a structural class of antibiotics with explainable deep learning

Save | Related Papers | Chat with paper

Felix Wong, Erica J. Zheng, Jacqueline A. Valeri, Nina M. Donghia, Melis N. Anahtar, Satotaka Omori, Alicia Li, Andres Cubillos-Ruiz, Aarti Krishnan, Wengong Jin, Abigail L. Manson, Jens Friedrichs, Ralf Helbig, Behnoush Hajian, Dawid K. Fiejtek, Florence F. Wagner, Holly H. Soutter, Ashlee M. Earl, Jonathan M. Stokes, Lars D. Renner & James J. Collins

Nature **626**, 177–185 (2024) | Cite this article

58k Accesses | **61** Citations | **1909** Altmetric | Metrics



La idea es explorar la síntesis de nuevas moléculas, las cuales pueden tener efectos beneficiosos.

- La representación de la molécula incluye los átomos que la componen (nodos) y los bonds (enlaces).

- Cada átomo y enlace tienen propiedades pero lo interesante ocurre al hacer el readout de la molécula, la cual se relaciona con un efecto.

Graph Transformer en ciencia

En la AF7 vamos a usar un benchmark que opera sobre las siguientes propiedades:

- Absorción.
- Distribución.
- Metabolismo.
- Excreción.
- Toxicidad.

Las moléculas vienen en el dataset en un formato conocido como SMILES:

Átomos:

Se representan por sus símbolos elementales (H, C, O, N, etc.).
Si un átomo tiene una carga o estado de oxidación específico, se escribe entre corchetes.

Enlaces:

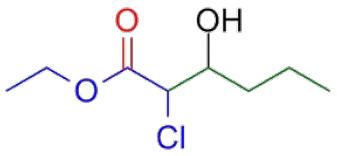
- Enlaces simples: No necesitan ser especificados, ya que se asume que todos los enlaces entre átomos son simples si no se dice lo contrario.
- Enlaces dobles: Se representan con =.
- Enlaces triples: Se representan con #.
- Enlaces aromáticos: Se indican usando letras minúsculas.

Ramas: Las ramas de una cadena principal se encierran entre paréntesis (). Ejemplo: CC(O)C para el propanol.

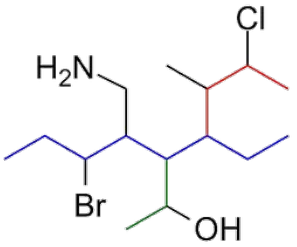
Anillos: Los átomos que forman parte de anillos se indican mediante números, que indican el inicio y el final del anillo. Ejemplo: C1CCCCC1 representa el ciclohexano.

Graph Transformer en ciencia

Ejemplo de molécula en formato SMILES:



CCOC(=O)C(C(O)CCC)Cl



CCC(Br)C(CN)C(C(O)C)C(C(C)C(Cl)C)CC

Ver esta referencia: medium

Para extraer características de la molécula se usa un extractor denomina RDKit. Luego, se mezclan tanto las características como la estructura para hacer regresión sobre un efecto ADMET:

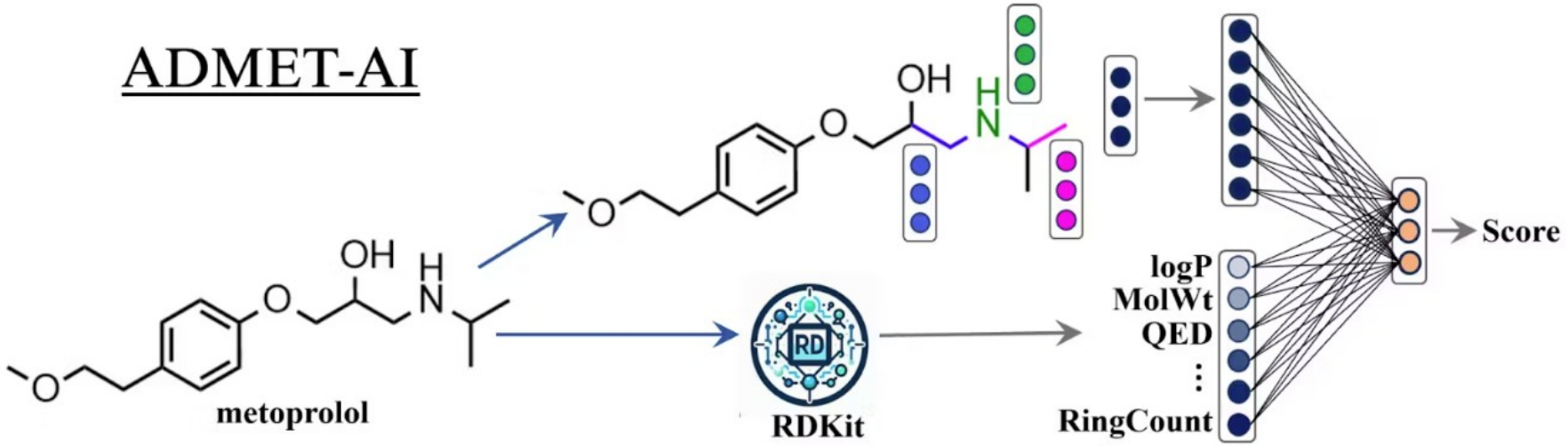


Figure 1. ADMET-AI uses machine learning models with the Chemprop-RDKit architecture. Chemprop-RDKit employs both a graph neural network (top) and 200 physicochemical properties computed by RDKit (bottom), which are combined by a feed-forward neural network (right) to predict the properties of a molecule.

Graph Transformer en ciencia

El benchmark ADMET viene separado por efectos y tiene distintos dataset de moléculas para cada tarea.

TDC		Current Top 1	
Task	Metric	Method	Score
Absorption			
Caco2	MAE	RDKit2D + MLP	0.393 ± 0.024
HIA	AUROC	AttrMasking	0.978 ± 0.006
Pgp	AUROC	AttrMasking	0.929 ± 0.006
Bioav	AUROC	RDKit2D + MLP	0.672 ± 0.021
Lipo	MAE	ContextPred	0.535 ± 0.012
AqSol	MAE	AttentiveFP	0.776 ± 0.008
Distribution			
BBB	AUROC	ContextPred	0.897 ± 0.004
PPBR	MAE	NeuralFP	9.292 ± 0.384
VDss	Spearman	RDKit2D + MLP	0.561 ± 0.025
Metabolism			
CYP2C9 Inhibition	AUPRC	AttentiveFP	0.749 ± 0.004
CYP2D6 Inhibition	AUPRC	AttentiveFP	0.646 ± 0.014
CYP3A4 Inhibition	AUPRC	AttentiveFP	0.851 ± 0.006
CYP2C9 Substrate	AUPRC	Morgan + MLP	0.380 ± 0.015
CYP2D6 Substrate	AUPRC	RDKit2D + MLP	0.677 ± 0.047
CYP3A4 Substrate	AUPRC	CNN	0.662 ± 0.031
Excretion			
Half Life	Spearman	Morgan + MLP	0.329 ± 0.083
CL-Hepa	Spearman	ContextPred	0.439 ± 0.026
CL-Micro	Spearman	RDKit2D + MLP	0.586 ± 0.014
Toxicity			
LD50	MAE	Morgan + MLP	0.649 ± 0.019
hERG	AUROC	RDKit2D + MLP	0.841 ± 0.020
Ames	AUROC	AttrMasking	0.842 ± 0.008
DILI	AUROC	AttrMasking	0.919 ± 0.008