GRAND: Graph Neural Diffusion Supplementary Material

Benjamin P. Chamberlain [*1] **James Rowbottom** [*1] **Maria Gorinova** [1] **Stefan Webb** [1] **Emanuele Rossi** [1]
Michael M. Bronstein [1 2]

1. Datasets

The statistics for the largest connected components of the experimental datasets are given in Table 1.

2. Diffusivity Formulations

GRAND can use any right stochastic attention matrix. We performed experiments with the multiheaded Bahdanau formulation (Bahdanau et al., 2014) of attention, which has previously been applied to graphs in (Veličković et al., 2018)

$$a(x_i, x_j) = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[W\mathbf{x}_i \| W\mathbf{x}_j\right]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[W\mathbf{x}_i \| W\mathbf{x}_k\right]\right)\right)}, \tag{1}$$

where W and $\mathbf{a}$ are learned and $\|$ is the concatenation operator. However, for all datasets, the scaled dot product attention performed better. This may be because GAT relies on dropout. Dropout performs poorly inside adaptive timestep numerical ODE solvers as the stochasticity in the forward pass drives $\tau \to 0$.

3. Full Training Objective

The full training program optimises cross entropy loss

$$\mathcal{L}(\mathbf{Y}, \mathbf{T}) = H(\mathbf{Y}, \mathbf{T}) = \sum_{i=1}^{n} \mathbf{t}_i^T \log \mathbf{y}_i \tag{2}$$

where $\mathbf{t}_i \in \mathbb{R}^{d_{\text{class}}}$ is the one-hot truth vector of the i^{th} node with prediction

$$\mathbf{y}_i = \psi(\mathbf{x}_i(T)) \tag{3}$$

where $\psi : \mathbb{R}^d \to \mathbb{R}^{d_{\text{class}}}$ is a linear layer decoder of the terminal value of the evolutionary PDE

$$\mathbf{y}_i = D\mathbf{x}_i(T) + \mathbf{b}_d \tag{4}$$

$$= D\left(\mathbf{X}(0) + \int_0^T \frac{\partial \mathbf{X}(t)}{\partial t} dt\right) + \mathbf{b}_d \tag{5}$$

$$= D\left(\phi(\mathbf{X}_{\text{in}}) + \int_0^T \frac{\partial \mathbf{X}(t)}{\partial t} dt\right) + \mathbf{b}_d \tag{6}$$

with the initial condition given by the linear layer encoder $\phi : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^d$ of the input data

$$\mathbf{y}_i = \left(D\left(E(\mathbf{X}_{\text{in}}) + \mathbf{b}_e + \int_0^T \frac{\partial \mathbf{X}(t)}{\partial t} dt\right) + \mathbf{b}_d\right)_i$$

and $f(\mathbf{X}(t), t, \theta) = \frac{\partial \mathbf{X}(t)}{\partial t}$ is the system dynamics that we wish to learn. For the nonlinear version of GRAND this is

$$f = \frac{\partial}{\partial t}\mathbf{X}(t) = (\mathbf{A}(\mathbf{X}(t)) - \mathbf{I})\mathbf{X}(t) = \bar{\mathbf{A}}(\mathbf{X}(t))\mathbf{X}(t)$$

4. Stability

4.1. Stability of linear ODE

In the main paper we reported the linear GRAND $\dot{\mathbf{x}} = \bar{A}\mathbf{x}$ has solution

$$\mathbf{x}(t) = \mathbf{x}(0)e^{\bar{A}t}. \tag{7}$$

As $\bar{A}$ is not diagonal this matrix exponential is not analytically recoverable. Performing eigenvalue decomposition the solution is

$$\mathbf{x}(t) = \bar{T}e^{\bar{D}t}\bar{T}^{-1}\mathbf{x}(0). \tag{8}$$

Assuming $\bar{T}^{-1}$ exists, $\bar{T}$ has full rank and both are bounded, the test equation becomes

$$\mathbf{y}(t) = e^{\bar{D}t}\mathbf{y}(0) \tag{9}$$

where $\mathbf{y}(t) = \bar{T}\mathbf{x}(t)$. If $\mathbf{x}(t)$ and $\hat{\mathbf{x}}(t)$ are two solutions of the ODE then their projections in eigenspace are $\mathbf{y}(t)$ and $\hat{\mathbf{y}}(t)$. For each node i:

$$|y_i(t) - \hat{y}_i(t)| = \left|(y_i(0) - \hat{y}_i(0))e^{\bar{\lambda}_i t}\right| \tag{10}$$

$$= |y_i(0) - \hat{y}_i(0)|e^{\mathcal{R}e(\bar{\lambda}_i)t} \tag{11}$$

for this to converge as $t \to \infty$ we require $\mathcal{R}e(\bar{\lambda}_i) \leq 0 \; \forall i$. As A is right stochastic the eigenvalues of $\bar{A} = A - I$ satisfy this property.

Dataset	Type	Classes	Features	Nodes	Edges	Label rate
Cora	citation	7	1433	2485	5069	0.056
Citeseer	citation	6	3703	2120	3679	0.057
PubMed	citation	3	500	19717	44324	0.003
Coauthor CS	co-author	15	6805	18333	81894	0.016
Computers	co-purchase	10	767	13381	245778	0.015
Photos	co-purchase	8	745	7487	119043	0.021
OGB-Arxiv	citation	40	128	169343	1166243	1

Table 1. Dataset Statistics

5. Numerical Schemes

5.1. Proof of theorem 1: Stability of explicit Euler

For linear GRAND with an Euler numerical integrator

$$\mathbf{x}^{(t+1)} = \left(I + \tau \bar{A}(\mathbf{x}^{(t)})\right) \mathbf{x}^{(t)} \tag{12}$$

$$= Q^{(t)} \mathbf{x}^{(t)}. \tag{13}$$

We require that the amplification factor $||Q^{(t)}|| < 1$. It is sufficient to show that $Q^{(t)}$ is a right stochastic matrix, which has the property that its spectral radius $\lambda_{\max} \leq 1$. Q is right stochastic if

1. $\sum_{j=1}^{N} q_{ij} = 1$
2. $q_{ij} > 0 \quad \forall i, j$

as A is right stochastic $\sum_j I_{ij} + \tau(A_{ij} - I_{ij}) = 1$ proving 1). As $a_{ij} = q_{ij}$ for $i \neq j$, to prove 2) it remains to show that $1 + \tau(a_{ii} - 1) > 0 \iff \tau < 1$.

5.2. Proof of theorem 2: Implicit methods

For implicit Euler

$$\dot{x}_n = \frac{x_n - x_{n-1}}{\tau} = f(x_n, t_n) \tag{14}$$

$$x_n = \tau f(x_n, t_n) + x_{n-1}, \tag{15}$$

incrementing the indices gives

$$x_{n+1} = \tau f(x_{n+1}, t_{n+1}) + x_n \tag{16}$$

and now, unlike the explicit case, x_{n+1} now appears on both sides of the equation. If f is linear

$$x_{n+1} = \tau \bar{A} x_{n+1} + x_n \tag{17}$$

$$x_{n+1} = (I - \tau \bar{A})^{-1} x_n = B^{-1} x_n = Q x_n, \tag{18}$$

and the matrix B must be inverted. The inverse exists as B is diagonally dominant

$$I_{ii} - \tau(A_{ii} - I_{ii}) > \tau \sum_{j \neq i} A_{ij} = \tau(1 - A_{ii}) \tag{19}$$

By considering the action of B on $w = (1, ..., 1)^T$ it is clear that $Bw = w \implies Qw = w \implies \sum_j Q_{ij} = 1$. As B is diagonally dominant it is irreducible and satisfies $B_{ij} \leq 0$ $i \neq j$ and $B_{ii} > 0$ giving $Q_{ij} > 0$ $\forall i, j$ (Varga, 1999) and Q is a Markov matrix with spectral radius bounded by unity and the implicit scheme is stable for all choices of τ.

6. General Multistep Methods

A general multistep method (combining both implicit and explicit methods) can be written as

$$x_{n+1} + \sum_{i=1}^{s} \alpha_i x_{n+1-i} = \tau \sum_{i=0}^{s} \beta_i f_{n+1-i}, \tag{20}$$

where $f = \dot{x}$. If $\beta_0 = 0$ then x_{n+1} only depends on terms up to n and the method is explicit.

6.1. Order

The order of a method gives the approximation error in terms of a Taylor series expansion. If p is the order, then the error is a single step $\propto \tau^{p+1}$ and the error in the entire interval $\propto \tau^p$. In practice the order of a numerical method can be determined by measuring how the error changes with step size for a known integral.

6.2. Butcher Tableau

The set of coefficients for each multi step method are given by the Butcher Tableau. The simple case of forward Euler has $\alpha_1 = -1$, $\beta_1 = 1$ with all other terms zero.

There is a law of diminishing return that relates the minimum number of function evaluations and the order of a higher order Runge-Kutta solver. Table 2 shows why the Runge-Kutta 4 method (RK4) is often regarded as the optimal trade-off between speed and accuracy for multi step solvers.

6.3. Runge-Kutta 4

For all experiments we find that Runge-Kutta 4 (or it's adaptive step size variants) outperforms lower order methods. The Runge-Kutta 4 method follows the schema: if

Order	1	2	3	4	5	6	7	8
Evals	1	2	3	4	6	7	9	11

Table 2. Function evaluations grow super-linearly with order after 4.

$$f(\mathbf{x}, t) = \bar{A}(\mathbf{x}_t)\mathbf{x}_t$$

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \frac{1}{6}\tau\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right) \tag{21}$$

$$\mathbf{k}_1 = f(\mathbf{x}_t, t) \tag{22}$$

$$\mathbf{k}_2 = f(\mathbf{x}_t + \tau\mathbf{k}_1/2, t + \tau/2) \tag{23}$$

$$\mathbf{k}_3 = f(\mathbf{x}_t + \tau\mathbf{k}_2/2, t + \tau/2) \tag{24}$$

$$\mathbf{k}_4 = f(\mathbf{x}_t + \tau\mathbf{k}_3, t + \tau) \tag{25}$$

6.4. Adaptive Step Size

Adaptive step size solvers estimate the error in x_{n+1}, which is compared to an error tolerance. The error is estimated by comparing two methods, one with order p and one with order $p-1$. They are interwoven, i.e., they have common intermediate steps. As a result, estimating the error has little or negligible computational cost compared to a step with the higher-order method.

$$x^*_{n+1} = x_n + \tau\sum_{i=1}^{s} b^*_i k_i \tag{26}$$

where k_i are the same as for the higher-order method. Then the error is

$$e_{n+1} = x_{n+1} - x^*_{n+1} = \tau\sum_{i=1}^{s}(b_i - b^*_i)k_i. \tag{27}$$

The time step is increased if the error is below tolerance and decreased otherwise.

7. Adaptive step size implementation details

Most results presented used the adaptive step size solver Dormand-Prince5. Key to getting this to work well is setting appropriate tolerances for the step size. Adaptive step size ODE solvers require two tolerance parameters; the relative tolerance rtol and the absolute atol. Both are used to assess the new step size

$$etol = atol + rtol * \max(|x_0|, |x_1|), \tag{28}$$

where x_0 and x_1 are successive estimations of the new state. Dupont et al. (2019) speculate that ResNets can learn a richer class of functions than ODEs because "the error arising from discrete steps allows trajectories to cross". We find that increasing the estimation error is also helpful when learning continuous diffusion functions and use value of $rtol$ and $etal$ that are $\times 10 - \times 1000$ larger than the defaults. This both improves prediction accuracy and reduces the runtime.

In hyperparameter search $atol$ and $rtol$ were paired together using a tolerance scale variable ts such that $atol = ts \times 10^{-12}$ and $atol = ts^{-6}$.

When using the adjoint method to backpropagate derivatives, two separate ODEs are being solved. This requires separate tolerance scales, which may differ: the forward pass tolerance, ts, controls for how close the approximated ODE solution is compared to the true solution, while the backward pass tolerance, ts_{adj}, controls the accuracy of the computed gradient. The hyperparameter search includes both ts and ts_{adj}.

References

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2014.

Dupont, E., Doucet, A., and Teh, Y. W. Augmented neural ODEs. In *NeurIPS*, 2019.

Varga, R. S. *Matrix Iterative Analysis*, volume 27. Springer Science & Business Media, 1999.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.