On graph classification, datasets & baselines

Enxhell Luzhnica*, Ben Day* and Pietro Lió
Department of Computer Science & Technology, University of Cambridge
Cambridge, United Kingdom.
el476@cam.ac.uk,ben.day@cl.cam.ac.uk

UNIVERSITY OF CAMBRIDGE

1. Introduction

Advances in graph coarsening have enabled the training of deeper networks and produced new state-of-the-art results in many benchmark tasks.

We examined how these architectures train and found that there is a **vanishing gradient problem** and that **performance is highly-sensitive to initialisation** and **depends strongly on jumping-knowledge (JK) structures**.

We then found very simple **sub-architectures** – structure-agnostic MLP, single-layer GCN and fixed-weight GCN – **to be competitive with the state-of-the-art.**

Preliminaries

The models we consider use the **improved GCN** (Gao and Ji, 2019) that double-weights the self-loop to bias localisation.

Jumping-knowledge structures (Xu et al., 2018) are a special case of skip connections where representations from many (often all) depths of the network are sent down the 'highway' to the final classifier. These were developed with the intention of allowing the network to use node representations from many 'ranges' (analogous to the receptive field).

top-k is a graph-coarsening operation that uses pruning to reduce the number of nodes, also from Gao and Ji (2019).

ReInit

Having found gradients vanishing at initialisation and later layers failing to be activated in trained models, we developed a data-driven initialisation to 'awaken' the model. The idea is related to LSUV (Mishkin and Matas, 2015) in that it works progressively layer-wise. First we initialise with any standard method and then proceed layer by layer, passing the entire training set and *standardising* the outputs

$$z(x) = \frac{x - \mu_x}{\sigma_x}$$
$$\mathbf{X}' = k \times z\big(\mathrm{GCN}(\mathbf{X}, \mathbf{A})\big)$$
$$\mathbf{X}'' = k \times z\big(\mathrm{X}'_{\vec{i}} \odot \tanh(\vec{y}_i)\big)$$

where the parameter k is introduced to control the spread, which may be necessary to avoid saturating the tanh or later activations. Clearly this results in zero-mean and variance k and working progressively we are able to ensure that later layers are active from the start of training.

References

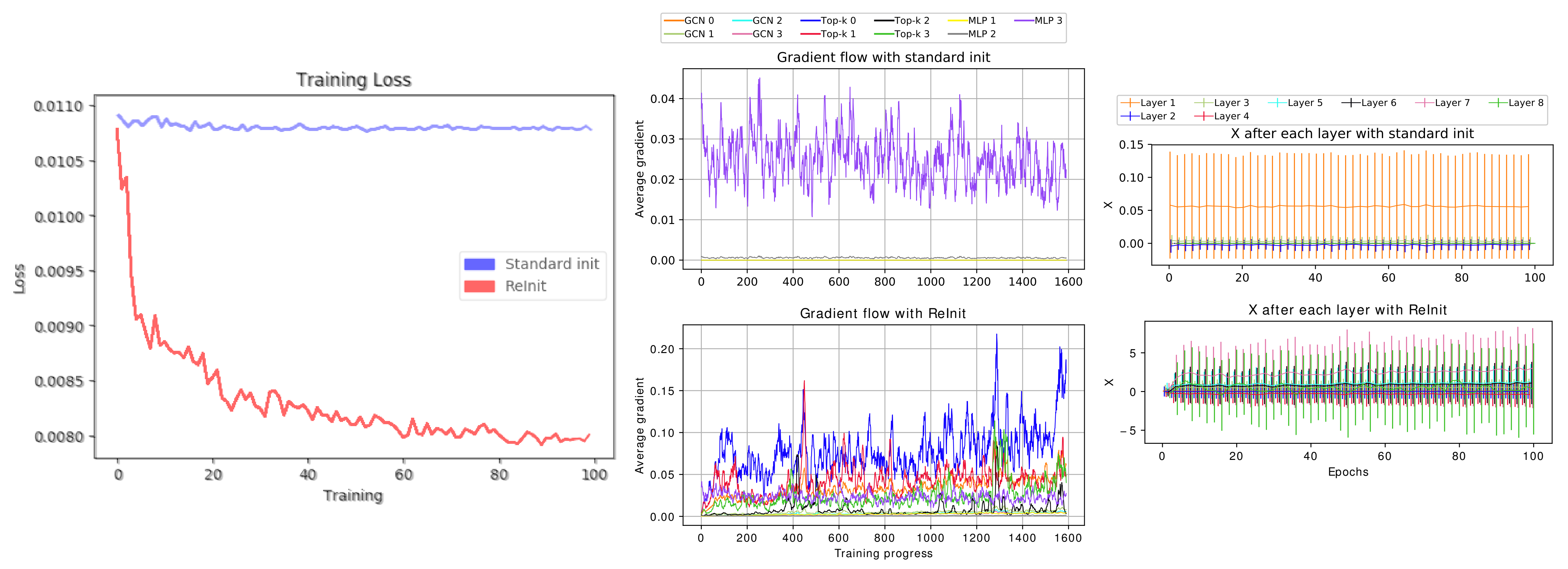
Cătălina Cangea, Petar Veličković, Nikola Jovanović, Thomas Kipf, and Pietro Liò. Towards Sparse Hierarchical Graph Classifiers. 11 2018. URL http://arxiv.org/abs/1811.01287.

Hongyang Gao and Shuiwang Ji. Graph U-Nets. 5 2019. URL http://arxiv.org/abs/1905.05178.

Dmytro Mishkin and Jiri Matas. All you need is a good init. 11 2015. URL http://arxiv.org/abs/1511.06422.

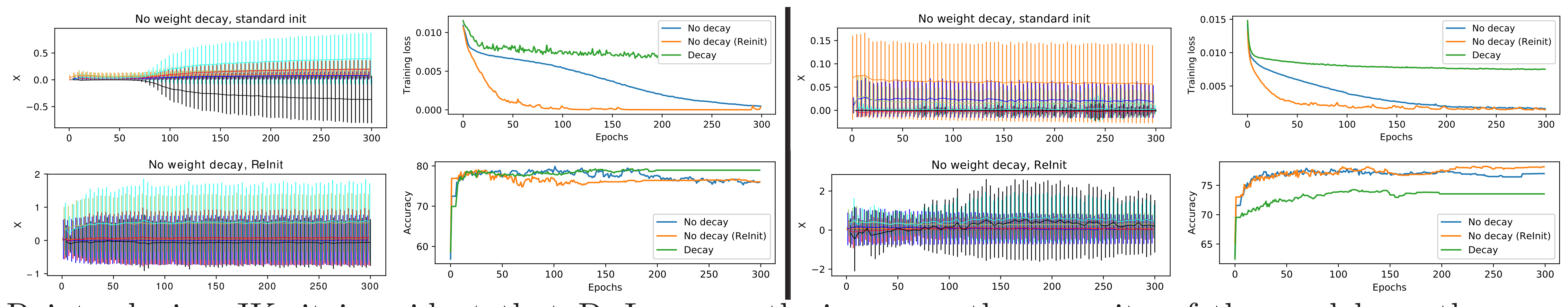
Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation Learning on Graphs with Jumping Knowledge Networks. 6 2018. URL http://arxiv.org/abs/1806.03536.

2. No JK



The vanishing gradient problem is made evident in this comparison using a 4-block GCN+top-k model. Removing JK structures renders deeper networks untrainable under standard initialisation. ReInit allows gradients to flow and training to proceed as usual.

3. ReInit and capacity



Reintroducing JK, it is evident that ReInit greatly increases the capacity of the models as they are now able to use the full extent of their depth. Predictably, this results in rapid overfitting on 'smaller' problems (DD, left) but performance is not improved on larger problems (Collab, right), indicating that the additional capacity is unnecessary. These tests use a 3-block GCN+top-k with MLP.

4. Results

Having found that JK is essential to training, that activations remain low in later layers throughout training under standard-initialisation and that activating deeper layers does not aid performance, we test the remaining possibility: the deeper networks are relying on shallow subarchitectures and using JK as a bypass only. We construct three sub-architectures to test this idea and find they perform at or near the state-of-the-art. We also test using JK with sum-mean readouts on a much larger problem (Reddit) where the additional capacity is found to be necessary.

	Datasets			
Model	Reddit-Multi-12K	DD	Collab	Proteins
Graphlet	21.73	74.85	64.66	72.91
Shortest-path	36.93	78.86	59.10	76.43
1-WL	39.03	74.02	78.61	73.76
WL-QA	44.38	79.04	80.74	75.26
PatchySAN	41.32	76.27	72.60	75.00
GraphSAGE	42.24	75.42	68.25	70.48
ECC	41.73	74.10	67.79	72.65
Set2Set	43.49	78.12	71.75	74.29
SortPool	41.82	79.37	73.76	75.54
DiffPool-Det	46.18	75.47	**82.13**	75.62
DiffPool-NoLP	46.65	79.98	75.63	77.42
DiffPool	47.08	**81.15**	75.50	**78.10**
GU-Net/SHGC	-	78.59	74.54	75.46
MLP (ours)	40.96	80.22	74.00	75.74
GCN(R)-MLP (ours)	36.15	78.61	75.38	76.28
GCN-MLP (ours)	45.01	79.29	76.50	75.64
JK-SM (ours)	**48.26**	78.77	75.92	75.82
JK-SM-Decay (ours)	47.75	79.11	74.14	75.82
JK-SM-ReInit (ours)	46.77	75.13	77.64	75.46

Table 1: Classification accuracy percentages. The results of other networks are taken from Cangea et al. 2018 with which we share 10-fold splits for benchmarking our methods. Bold indicates top-performance, blue and red indicate weaker performance than the structure-agnostic MLP and GCN-MLP, respectively.