Deep Topology Classification: A New Approach for Massive Graph Classification

Stephen Bonner, John Brennan, Georgios Theodoropoulos, Ibad Kureshi and Andrew Stephen McGough

School of Engineering and Computing Sciences
Durham University, Durham, UK

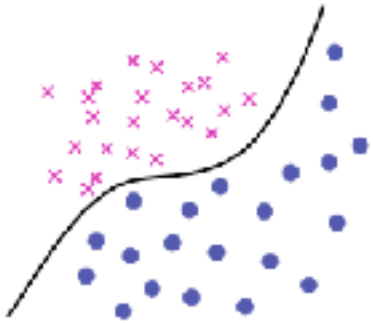
Durham University

NVIDIA

EPSRC
Engineering and Physical Sciences Research Council

Motivating Examples

- **Graph Classification:** Graph classification is key area within the field of network science with many applications across the scientific disciplines.
- Graph classification can broadly be split into two different branches:
- **Within Graph Classification -** The classification of individual elements within a graph. Often used for link prediction and product recommendations, as such is widely studied.
- **Global Graph Classification -** Used to classify the entire graph as belonging to a certain class. Approaches can be based on labels or on topological structure of the graphs. Could be used for identification of chemical compound or identification of users based upon their complete social network graph.



Global Graph Classification

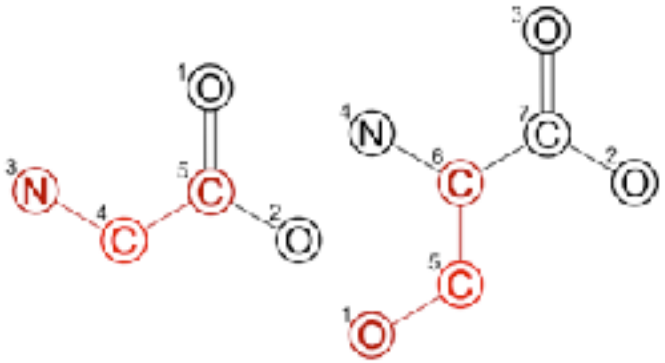
- **Global Graph Classification:**

- In this problem we have a dataset - $\mathcal{D}$
- This comprises of N graphs $G_i \in \mathcal{D}$ where $i = 1, ..., N$ and $G_i = (V_i, E_i)$.
- Each graph in $\mathcal{D}$ has a corresponding class $y_i \subset \mathcal{C}$ where $\mathcal{C}$ is the set of k categorical class labels, given as $\mathcal{C} = 1, ..., k$.

- The goal of the global graph classification task is to derive a mathematical function to perform $f : \mathcal{D} \rightarrow \mathcal{C}$
- When deriving f using a machine learning approach, the common pattern is to learn the function from a subset of $\mathcal{D}$ known as the training set for which labels are present.
- The function is then tested on the remaining examples from $\mathcal{D}$, often called the test set.
- The accuracy of the function is assessed by comparing the predicted label $\hat{y}_i = f(G_i)$ with the ground truth label (y_i) for all graphs in $\mathcal{D}$
- Problem is that many ML models for classification require an N-dimensional vector as input - Will not take graphs as raw input.

Previous Work

- **Previous Approaches -**
- **Graph Kernels:** Many of the existing approaches for graph classification are based on graph kernels.
- Sub-graph kernels are particularly popular for the global classification problem but other approaches have been used as well. Still concerns about scalability if the dataset is large.
- Common approach is to use graph kernels as features, with an SVM to perform the classification.



X	1	2	3	4	5	6	7
1	.50	.50	.98	.00	.00	.00	.00
2	.89	.98	.50	.34	.17	.16	.11
3	.38	.33	.00	.91	.20	.13	.14
4	.20	.24	.00	.17	.77	.81	.67
5	.13	.11	.00	.14	.78	.68	.96

Motivating Examples

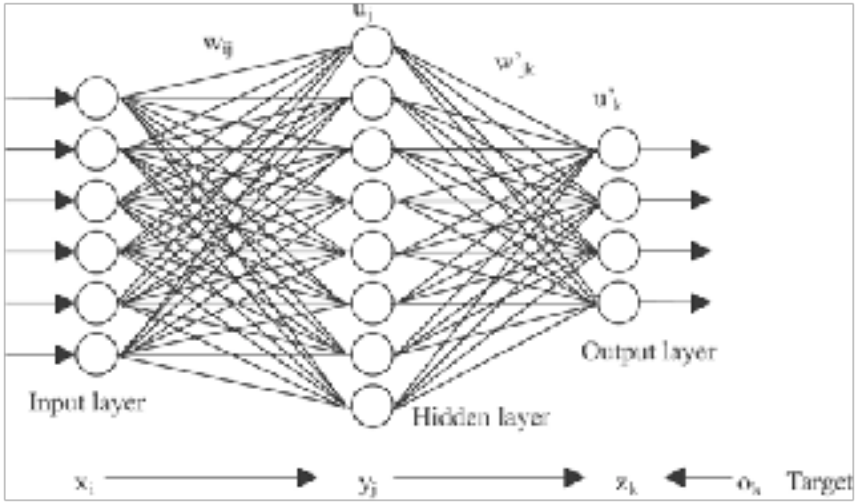
- **Previous Approaches -**

- **Feature Extraction Methods:** Comparatively much fewer approaches which explore the use of topological features for global graph classification.
- Most comprehensive approach is presented by Li'12. In which they use a variety of global topological features and an SVM for classification.
- Across a range of graphs, their approach GF is consistently more accurate than a range of state of the art graph kernels.
- Also shown to be much quicker to compute.

	EB vs. EI	EN vs. EB	EN vs. EI	OF vs. ON	ON vs. OO
GF(no)	57.78 ± 7.25	61.92 ± 5.21	64.48 ± 4.96	65.67 ± 20.76	71.19 ± 14.06
GF(r)	87.70 ± 5.30	**86.35 ± 7.02**	82.76 ± 5.35	**98.33 ± 5.00**	**94.29 ± 7.00**
GF(z)	**88.05 ± 4.27**	84.58 ± 6.96	**83.84 ± 4.41**	97.67 ± 6.67	92.86 ± 7.14
RW	–	–	–	90.00 ± 11.06	76.43 ± 14.30
SP	76.27 ± 5.16	77.61 ± 6.29	73.22 ± 6.14	94.67 ± 8.19	78.33 ± 14.57
GK	66.01 ± 9.57	75.19 ± 8.13	62.38 ± 6.91	56.00 ± 21.12	60.71 ± 13.27
RG	–	–	–	72.33 ± 12.52	67.04 ± 15.54
WL	87.23 ± 4.57	74.81 ± 6.09	71.22 ± 8.15	**98.33 ± 5.00**	63.57 ± 17.63
t-statistic	0.23	9.31	**12.74**	0	**12.28**

Approach Overview

- **We aim to explore the use of graph feature vectors as a way of classifying graphs.**
- Inspired by work by Li, we want to expand upon it by the inclusion of vertex level feature as well as global features.
- Inspired by recent developments in within graph classification, we create a deep feed forward for the classification, rather than the traditional use of SVMs.



Approach Overview

- **Using the GFP feature vectors.**

- *Graph order and number of edges*
- *Number of triangles*
- *Global clustering coefficient*
- *Maximum total degree*
- *Number of components.*

$$VF_{n,m} = \begin{pmatrix} f_{1,2} & \cdots & f_{1,n} \\ f_{2,2} & \cdots & f_{2,n} \\ \vdots & \ddots & \vdots \\ f_{m,2} & \cdots & f_{m,n} \end{pmatrix}$$

Local Features:

- *Eigenvector Centrality Value*
- *PageRank Value*
- *Total Degree*
- *Number Of Two HopAway Neighbours*
- *Local Clustering Score*
- *Average Clustering of Neighbourhood*

$$\overrightarrow{V_G 1} = (\bar{x}_1, Mo_1, \sigma_1, \sigma_1^2, Skew[x]_1, Kurt[x]_1, x(1)_1, x(n)_1, \ldots$$
$$, \bar{x}_n, Mo_n, \sigma_n, \sigma_n^2, Skew[x]_n, Kurt[x]_n, x(1)_n, x(n)_n)$$

Approach Overview

- **ANN Model Creation:**
- There are a large number of choices that must be made when designing a neural network.

- We performed a grid search over many of the common choices in the literature from neurone initialisation and activation strategies to the number of hidden layers and units to create our network.

- Created two version of the *DTC* network, one for binary and multi class classification.

DTC NETWORK ARCHITECTURE

Layer	Size	Initialisation	Activation	Dropout
Input	$\lvert\overrightarrow{F_i}\rvert$	NA	NA	NA
First Hidden	256	Glorot-Uniform	ReLU	0.2
Second Hidden	128	Glorot-Uniform	ReLU	0.2
Third Hidden	32	Glorot-Uniform	ReLU	0.2
Multi-Output	$\lvert C\rvert$	NA	Softmax	NA
Binary-Output	1	NA	Sigmoid	NA

Background Technologies

- **We extract the features using the Spark, then perform the classification using TensorFlow and Keras.**

- **Apache Spark** - An in memory computation layer for the Hadoop ecosystem. It has a variety of domain-specific computation libraries including GraphX, Spark Streaming, MLlib and SparkSQL.
- **TensorFlow** - An open source software library for numerical computation using data flow graphs created by Google. Enables the easy use of GPU computing.
- **Keras** - A deep learning library which sits on top of TensorFlow and provides several preconfigured ANN layers.

Experimental Evaluation and Results

- **Hardware:**
- Software stack of CentOS 7.2, CUDA 7.5, CuDNN v4, TensorFlow 0.10.0 and Keras 1.0.8.
- Tested on a single node with 2 Nvidia Tesla K40c's, 20C 2.3GHz Intel Xeon E5- 2650 v3, 64GB RAM
- **Testing Methodology:**
- All the accuracy scores presented are the mean accuracy after k-fold cross validation.

Experimental Evaluation and Results

- **Datasets:**

- ANN's require large quantities of massive graph datasets, due to this we use synthetic generated graphs for this work.
- *One future research direction is to explore augmentation and sampling techniques on network datasets to enhance the high quality existing network repo's such as SNAP and The Network Repository.*

- **Dataset One (Multi-Class):** 50,000 graphs in total, with 10,000 from each of the following generation methods: Forest Fire, Barabasi-Albert, Erdos-Renyi, R-MAT and Small World. Where required we randomised the parameters to avoid overfitting to one set.

- **Dataset Two (Binary):** 20,000 graphs in total, with 10,000 Forest Fire graphs and 10,000 'rewired' graphs. Number of rewired edges chosen uniformly at random from between 100 to 10,000.

Experimental Evaluation and Results

- 10 Fold classification results for the multi-class dataset:

MULTI-CLASS CLASSIFICATION RESULTS

Method	Accuracy (%)	Recall	Precision	F1 Score
DTC (Scaled)	**99.958 ± 0.074**	**0.99998 ± 0.00004**	**0.99998 ± 0.00004**	**0.99998 ± 0.00004**
DTC (Unscaled)	70.443 ± 7.819	0.70497 ± 0.07782	0.71247 ± 0.07862	0.70870 ± 0.012931
SVM (Full-Scaled)	88.432 ± 1.100	0.88396 ± 0.00867	0.88426 ± 0.00867	0.884261 ± 0.01097
SVM (Full-Unscaled)	26.113 ± 0.501	0.26079 ± 0.00948	0.25925 ± 0.00501	0.25897 ± 0.00721
SVM (Global-Scaled)	54.483 ± 1.252	0.54451 ± 0.01378	0.54483 ± 0.01252	0.54487 ± 0.01401
SVM (Global-Unscaled)	50.673 ± 1.092	0.50631 ± 0.01301	0.50673 ± 0.01092	0.50681 ± 0.01418

- Comparing with an SVM to replicate the Li approach.

Experimental Evaluation and Results

- These figures highlight the error matrices for the different approach one dataset one.

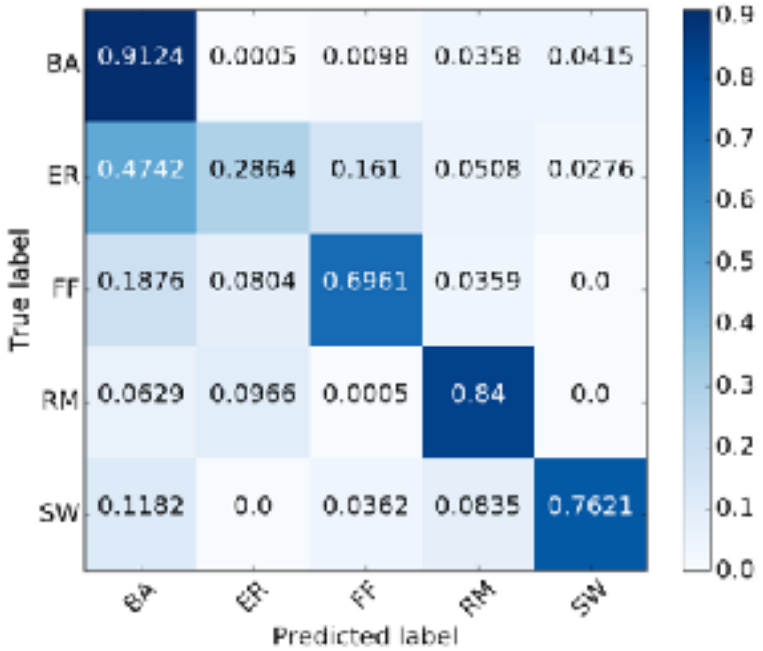


Figure 1. Normalized Error Matrix For SVM (Scaled)

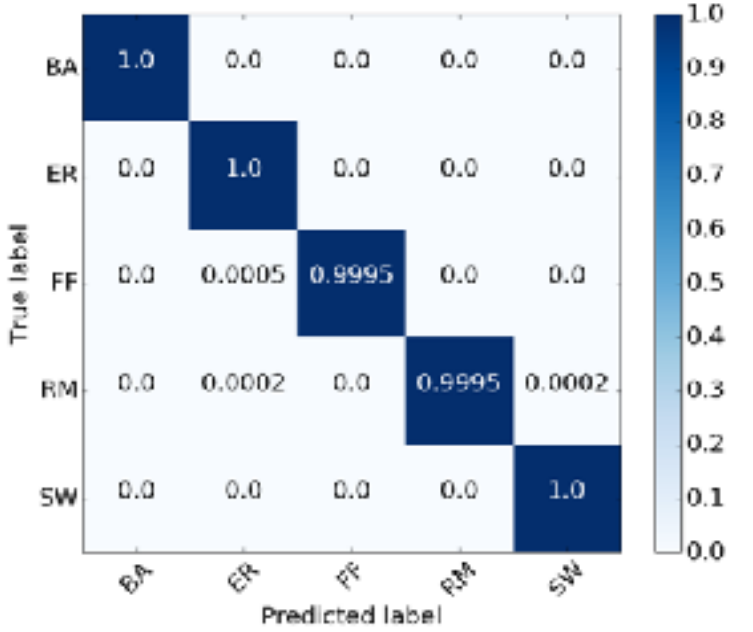


Figure 2. Normalized Error Matrix For DTC (Scaled)

Experimental Evaluation and Results

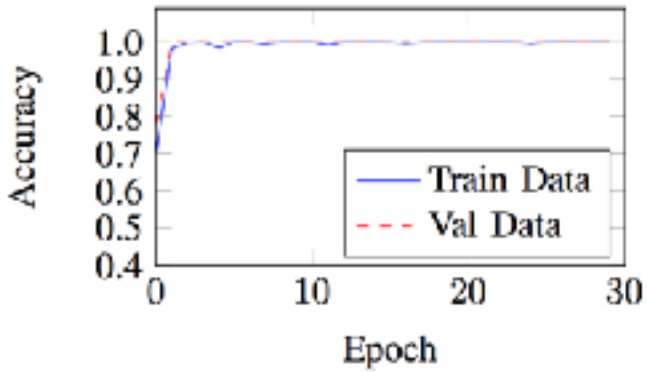
- 10 Fold classification results for the binary dataset:

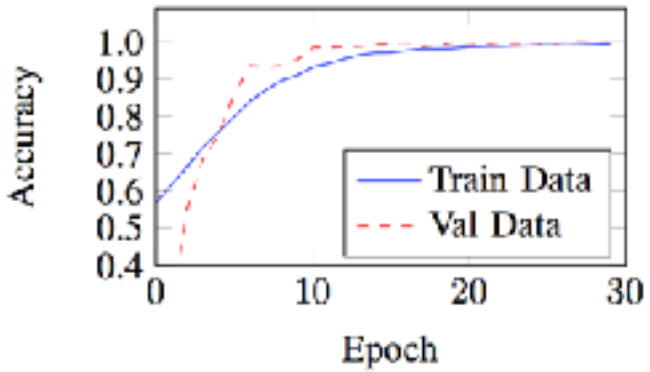
BINARY CLASSIFICATION RESULTS

Method	Accuracy (%)	Recall	Precision	F1 Score
DTC (Scaled)	**99.980 ± 0.049**	**0.99995 ± 0.00015**	**0.99995 ± 0.00015**	**0.99995 ± 0.00015**
DTC (Unscaled)	51.435 ± 8.793	0.48850 ± 0.00983	0.52710 ± 0.00983	0.507066 ± 0.33614
SVM (Full-Scaled)	68.034 ± 8.821	0.70012 ± 0.31304	0.68034 ± 0.28739	0.71509 ± 0.33614
SVM (Full-Unscaled)	49.045 ± 1.141	0.48910 ± 0.01566	0.49045 ± 0.01141	0.49145 ± 0.00929
SVM (Global-Scaled)	56.482 ± 13.435	0.56780 ± 0.13913	0.57834 ± 0.14034	0.57302 ± 0.13024
SVM (Global-Unscaled)	42.546 ± 2.914	0.42916 ± 0.02959	0.43813 ± 0.03152	0.43359 ± 0.03102

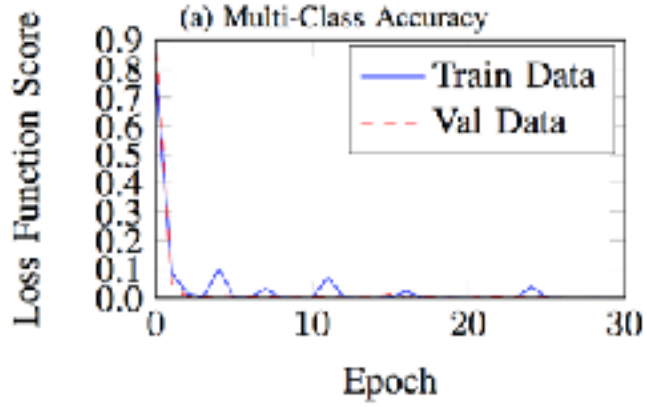
Experimental Evaluation and Results

- **Classification accuracy over training epochs:**
- Interesting to note that the train and validation accuracy curves match, show that the model is not overfitting to the training data.
- Also shows that the binary classification task is the more complicated of the two tasks due to the increase number of epochs required.

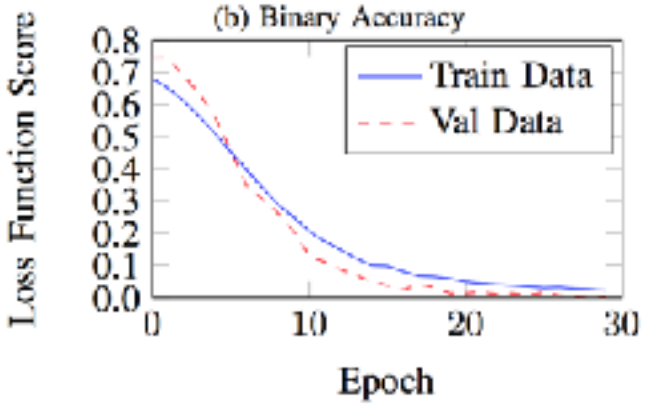




(a) Multi-Class Accuracy



(b) Binary Accuracy



Conclusions and Further Work

- **Conclusions:**

- Introduced the DTC approach for massive graph classification.
- Uses a combination of local and global graph features, classified via a deep ANN.
- Beats the current state of the art approach on two synthetic graph datasets.

- **Future Work:**

- Move to a complete custom TensorFlow implementation.
- Compare more throughly with existing Graph Kernel based methods.
- Move to testing on real benchmark datasets exploring the use of Network Sampling techniques. Possible application in graph based anomaly detection.
- Begin testing with unbalanced datasets

Please note that all code and experiment scripts are open sourced under GPLv3 and is available on GitHub -

https://github.com/sbonner0/DeepTopologyClassification