# Deep Topology Classification: A New Approach for Massive Graph Classification

Stephen Bonner, John Brennan, Georgios Theodoropoulos, Ibad Kureshi, Andrew Stephen McGough
*School of Engineering and Computing Sciences*
*Durham University, Durham, UK*
{*s.a.r.bonner, j.d.brennan, georgios.theodoropoulos, ibad.kureshi, stephen.mcgough*}*@durham.ac.uk*

***Abstract*—The classification of graphs is a key challenge within scientific fields using graphs to represent data and is an active area of research. Graph classification can be critical in identifying and labelling unknown graphs within a dataset and has seen application across many scientific fields. Graph classification poses two distinct problems: the classification of elements within a graph and the classification of the entire graph. Whilst there is considerable work on the first problem, the efficient and accurate classification of massive graphs into one or more classes has, thus far, received less attention.**

**In this paper we propose the *Deep Topology Classification (DTC)* approach for global graph classification. *DTC* extracts both global and vertex level topological features from a graph to create a highly discriminate representation in feature space. A deep feed-forward neural network is designed and trained to classify these graph feature vectors. This approach is shown to be over 99% accurate at discerning graph classes over two datasets. Additionally, it is shown more accurate than current state of the art approaches both in binary and multi-class graph classification tasks.**

***Keywords*-Deep Learning; Graph Classification;**

## I. Introduction

Representing data as graphs or networks has enabled researchers to not only understand the data itself but also any underlying relationships. This approach to data analysis is being used by fields as disparate as archaeology to biology. Being able to accurately match a graph, which may not have complete descriptive information, to its domain and application can help to identify unknown data. As such, there has been increasing interest in the literature on how best to develop models to classify these graph datasets [1] [2]. There are two different types of graph classification; classifying individual elements (vertices or edges) within a graph and classifying the entire discrete graph itself. In this paper, we are considering the second of these two problems; global graph classification. Global graph classification is required for a myriad of tasks within the field of network analysis (e.g. protein identification, social circle identification). The terms graph and network are often used interchangeably within the literature, however, we shall use the term graph without loss of generality.

The volume of graph data, both in terms of size and complexity of individual graphs and the total number of available graphs, is increasing rapidly [3]. The current Facebook social network graph, for example, is said to contain over one billion vertices and is still growing [4]. Traditionally, graph classification has been performed by the use of graph kernel methods [5], but such methods can take a prohibitively long time to compute, even on comparatively small graphs of a few thousand vertices. Hence making the applicability of graph kernels questionable on modern massive graphs.Thus, a new approach to massive graph classification is needed which does not require the use of graph kernel methods.

In this paper we present a novel approach for both multi-class and binary classification of massive complex graphs entitled *Deep Topology Classification (DTC)*. *DTC*, unlike previous approaches, extracts both global and deep topological features from each graph to transform it into a feature space. To perform the classification itself, a deep neural network is designed and trained. The approach is shown to be more accurate than the current state of the art topological feature graph classification method. The key contributions of this paper are:

- *Novelty* - The *DTC* approach is, to the best of our knowledge, the first in the literature to make use of a deep neural network for global graph classification and to enable both multi-class and binary classification to be performed.
- *Accuracy* - The *DTC* approach is shown to be more accurate than state of art feature extraction methods.
- *Reproducibility* - The code and datasets used for the approach and results presented in this paper have been open-sourced on a GPLv3 licence and are available online[1]. In addition, we have included the pre-trained neural network for complete reproducibility.

In Section 2 we define the classification problem, Section 3 highlights previous research, Section 4 introduces the DTC method, Section 5 details the experimentation performed, Section 6 presents the results and Section 7 draws conclusions and presents scope for future research.

## II. Problem Definition

We are considering the problem of classifying graphs into their correct categories via the use of machine learning. We define a graph $G = (V, E)$ as a finite set of vertices (sometimes referred to as nodes) – $V$ – and a finite set of edges – $E$. The elements of $E$ are an unordered tuple $\{u, v\}$ of vertices $u, v \in V$. In this problem, we have a dataset

[1] https://github.com/sbonner0/GraphFingerprintComparison

$\mathcal{D}$ comprising $N$ graphs $G_i \in \mathcal{D}$, where $i = 1, ..., N$ and $G_i = (V_i, E_i)$ where a label might be present on the vertices or edges, although the work considered in this paper requires no such labels. Each graph in $\mathcal{D}$ has a corresponding class $y_i \in \mathcal{C}$, where $\mathcal{C}$ is the set of $k$ categorical class labels, given as $\mathcal{C} = 1, ..., k$. In the case of graphs, the categorical class label could correspond with a graphs domain, for example a social, biological or citation network, or the synthetic generation method used.

The goal of the global graph classification task is to derive a mathematical formulae to perform $f : \mathcal{D} \rightarrow \mathcal{C}$ which can accurately predicate the class label of each graph in the dataset. If 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 for all graphs in $\mathcal{D}$. Established models for classification, such as Support Vector Machines (SVMs), Decision Trees or Artificial Neural Networks (ANNs) do not function with graphs as these models require an $\mathcal{N}$-Dimensional vector as input. Therefore before any graph can be passed to the function, its inherent discrete nature must first be converted into a vector. Due to the size and complexity of modern graphs, this is considered one of the most challenging aspects of global graph classification [6].

## III. Previous Work

The field of graph classification can be divided into two major categories; within graph classification and global graph classification. Within graph classification encompasses techniques designed to classify individual vertices or edges within a single graph and has been extensively explored by prior work [7]. Global graph classification techniques attempt to classify the type or domain of an entire graph. However, there is comparatively less research focusing on the classification of the entire graph, perhaps owing to the lack of sufficient quantity of publicly available datasets and the complexity of discovering an appropriate vector representation.

### A. Within Graph Classification

A selection of recent work on vertex classification has explored the use of a single hidden-layer neural network to learn features required for classification in an unsupervised manner. These approaches are inspired by the *word2vec* [8] or *SkipGram* [9] methods for automated feature learning from text documents, adapting the technique to fit graph data. The *DeepWalk* approach [2] uses a random walk to sample the structure of a vertexs neighbourhood which is then fed into the *SkipGram* model, with the sequence of vertices replacing the sequence of words within a sentence. *DeepWalk* has been shown to be more accurate at classifying vertices in a variety of datasets than traditional methods like *SpectralClustering* [10] and *EdgeClustering* [11]. The *node2vec* [7] approach expands upon this method by having a flexible definition of a vertices neighbourhood. This is achieved by biasing the random walk to explore the vertices close or far from a given vertex, leading to a greater understanding of its local or global role within a graph.

### B. Global Graph Classification

**Graph Kernel Methods** - A large body of work has been performed to classify graph datasets based upon graph kernels, a series of kernel functions which compute an inner product on graphs. Graph kernels include random walks [12], shortest path [13] and discriminative subgraphs [14]. Readers are referred to a survey on graph kernels for a greater discussion on their uses outside of classification [5]. Sub-graph kernels (sub-graphs which are found frequently in a given graph) are perhaps the most explored for performing graph classification [15] [16]. In such approaches frequent discriminative sub-graphs are mined using a variety of kernel techniques and used as features for classification. Work has shown that larger subgraphs result in a more accurate classification but at the cost of a greatly increased runtime for feature extraction [15]. The use of sub-graph kernels for classification has been further explored when considering noisy and unbalanced datasets [17]. Graph kernels have been explored for multi-label classification of graph datasets in an approach entitled *gMLC* [18]. The *gMLC* approach uses an SVM to train a model to assign one or more labels from a set of possible labels to a range of medical and biological graphs. Graph kernels have also been utilised as a way to classify streams of massive time-series graph datasets in a memory efficient manner [19]. The approach uses the Weisfeiler-Lehman graph kernel [20] and an SVM in an incremental manner. To reduce the large memory footprint inherent in graph streams, the support vectors from the previous time steps are used as training data for the model.

Graph kernels are known to be prohibitively to slow extract from large graphs [7], thus are not suitable for our approach as we are attempted to classifying massive graphs.

**Topological Feature Methods** - There are a few approaches to graph classification which employ the extraction of topological features rather than the use of graph kernels. These approaches are designed to overcome the inherent problems of scalability and runtime efficiency required when extracting graph kernels [6]. There are numerous features which can be extracted from graphs and the technique has been used successfully for many graph mining tasks including graph similarity measuring [21], time series anomaly detection [22] and link prediction [23].

Work has been performed to explore the application of topological graph features to differentiate between graphs from different domains [24]. Although the work stops short of creating an actual classifier, it does conclude that both local and global features can be useful in identifying a graph's domain [24].

Li et al. propose a novel method of classifying graphs into domains based on the extraction of global and label based features [1] [6]. The approach uses an SVM to classify the resulting feature vectors. The features are scaled using both range normalisation and z-normalisation to overcome the different scale of chosen features. The work presents results on the classification of three different graph datasets including chemical compound graphs, protein graphs and cell graphs. The approach is shown to be more accurate than state of the art graph kernel based methods [1]. However, the approach is not extended to datasets in which multiple classes may be present and is missing the potentially rich descriptive features at the vertex level.

## IV. DTC Approach

*Deep Topology Classification (DTC)* is an approach to the classification of massive complex graph datasets. *DTC* extracts both global and deep topological features from a given graph, rather than using a graph kernel method for feature representation. We take inspiration from research showing how the use of global topological features can be used to beat the classification accuracy of graph kernel based methods [6]. The *DTC* approach further improves upon this by exploring the use of deep topological features, rather than just global metrics. An additional benefit over the previous graph kernels based approaches is that our feature extraction procedure can be completed in approximately $O(n)$ [21]. To classify the resulting vector representations, a deep feed-forward neural network is created and trained. The use of a deep neural network, rather than the traditional SVM utilised in the global graph classification literature, was inspired by recent advances in within-graph classification using neural networks [7].

Below we detail the feature extraction stage and the design of the deep neural networks used by *DTC*.

### A. Feature Extraction

In order to perform any classification, the graphs must be transformed into a set of features via a feature extraction process. The *DTC* approach extracts a selection of deep topological features from each graph it is classifying. The use of topological features was required when we found that graph kernel and embedding based approaches have, in the best case, a third order polynomial execution time [20]. Since one of the key requirements of the *DTC* approach is that it is able to classify potentially thousands of massive graphs, graph kernel and embedding approaches are not suitable due their computational complexity. In order to ensure the most accurate classification features which are extremely discriminative are required. In previous work, we introduced a graph feature extraction approach called *Graph Fingerprinting - GFP* which was shown to be highly accurate at detecting similarities between graphs with nearly identical topologies [21]. In this work, we explore the suitability of the *GFP* feature vectors for the task of global graph classification.

The *GFP* approach extracts a series of global and deep topological features from each graph in a given dataset. The process is explained in detail in previous work [21] but we will briefly review it here. Each graph $G_i \in \mathcal{D}$ is passed to the *GFP* approach which extracts a corresponding feature vector $\overrightarrow{F_i}$, containing both global and local features. Local features are extracted for each $v \in V$, as detailed in Table I. All these features are combined into a matrix of features of size $|V| * |Q|$, where $Q$ is the number of local features. To aggregate this down, the following metrics are extracted for each column in the feature matrix: median $\bar{x}_1$, mean $Mo_1$, standard deviation $\sigma_1$, variance $\sigma_1^2$, skewness $Skew[x]_1$, kurtosis $Kurt[x]_1$, minimum value $x(1)_n$ and maximum value $x(n)_n$.

In addition the global features detailed below are appended to $\overrightarrow{F_i}$:

**Graph Order** - Defined as: $|V|$.

**Number of Edges** - Defined as: $|E|$.

**Number of Triangles** - The number of triangles, $\alpha$, is the number of vertices which form a triangle, with a triangle being a set of three vertices with an edge between every member.

**Maximum Total Degree Value** - The total number of edges the most connected vertex in the graph has to other vertices.

**Number of Components** - The total number of components within the graph, with a component being a subgraph in which there is a possible path between every vertex, whilst vertices in different components have no possible path between them.

**Global Clustering Coefficient** - The total number of possible vs complete triangles within a graph: $gc = 3\alpha/\beta$, where $\beta$ is the number of connected triplets of vertices (three vertices which are all connected, but not necessarily into a triangle) within the graph.

Many machine learning models require that the features are standardised to the same range, often this is achieved via the use of scaling [6]. For the *DTC* approach, the feature vector $\overrightarrow{F_i}$ is scaled to have zero mean and unit variance.

### B. Model Design

Artificial Neural Networks (ANNs) and Deep Learning are fields within machine learning inspired by the functionality of the human brain [30]. ANNs model problems via the use of connected layers of artificial neurons. Each ANN has an input layer, at least one hidden layer and an output layer. The activation of each neuron is given by a pre-specified function, with each neuron taking a weighted sum of the outputs of the neurons connected to it. The weights of the network are 'learned' as training examples are fed through the network that generate output.

Table I
DEEP TOPOLOGICAL FEATURES

| Feature Name | Equation |
|---|---|
| **Eigenvector Centrality Value** - Used to calculate the importance of each vertex within a graph. Eigenvector centrality can be written as an eigenvector equation, where $\lambda$ is the largest eigenvalue, $\mathbf{A}$ is the graph in adjacency matrix form and $\mathbf{x}$ is the eigenvector centrality [25]. | $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$. |
| **PageRank Score** - The PageRank centrality is commonly used to measure the local influence of a vertex within a graph [26] [27]. Where $\Gamma^{-}(v)$ is the set of incoming neighbours of $v$, $d^{+}(u)$ is the out-degree of $u$ and $d$ is a constant damping factor (0.85 for this work). | $PR(v) = \frac{1-d}{N} + d \sum_{u \in \Gamma^{-}(v)} \frac{PR(u)}{d^{+}(u)}$ |
| **Average PageRank of Neighbourhood** - The average PageRank of the neighbourhood is the mean of each PageRank score within a vertices neighbourhood, where $PR(j)$ is the PageRank score calculated in the previous step and $N(v)$ is every vertex incident on the current vertex $v$ [21]. | $N_{PR(v)} = \frac{1}{\lvert N(v) \rvert} \sum_{\forall j \in N(v)} PR(j)$ |
| **Total Degree** - This is the sum of both the in and out degree for the vertex $v$ [21]. | $td_v = \Gamma^{-}(v) + d^{+}(v)$ |
| **Two-Hop Away Neighbours** - The number of two-hop away neighbours from the current vertex $v$ gives an indication of how connected a vertex's neighbourhood is within the graph [28]. | $th_v = \frac{1}{\lvert N(v) \rvert} \sum_{\forall j \in N(v)} d^{+}(j)$ |
| **Local Clustering Score** - Represents the probability of two neighbours of $v$ also being neighbours of each other, where $\Phi$ is the number of pairs of $v's$ neighbours which are themselves connected [29]. | $c_v = \frac{2\Phi}{d^{+}(v)(d^{+}(v)-1)}$ |
| **Average Clustering of Neighbourhood** - The mean of all the local clustering scores for the vertex's neighbourhood, where $c_j$ is the local clustering score [28]. | $nc_v = \frac{1}{\lvert N(v) \rvert} \sum_{\forall j \in N(v)} c_j$ |

Modifications are made to the weights in the network to increase the probability of producing the desired outcome. The weights are updated through the use of back-propagation [31]. Deep Learning is a term generally used to refer to ANN's which have multiple hidden layers. Types of Deep ANN include Deep Feed Forward (used often for logistical classification), Convolutional Neural Networks (which can use an image as input as thus is used for tasks such as image classification) and Recurrent Neural Networks (which can consider a sequence of input vectors and thus can be used for time series analysis). Readers interested in the different types of Deep Learning highlighted above are referred to [30]. Deep Learning has been extremely successful in solving complicated non-linear problems in computer vision and natural language processing. However, so far the application of Deep Learning within the field of graph processing has been limited [7] [2].

The ANN we have created for the *DTC* approach follows the Deep Feed Forward (DFF) model, meaning that each layer in the network is fully connected with the subsequent layer. The size of the input layer is equal to the dimensionality of the extracted GFP vector and the size of the output layer being equal to the number of unique categorical class labels. When designing a neural network, several key choices must be made in regards to the number of hidden layers, the random initialisation of the neuron's weights and the activation function they use. In addition a suitable loss function, a function which the ANN is trying to minimise, must be chosen to ensure the most accurate model. To select the correct functions and parameters for the *DTC* network, we performed a grid search over a selection of well regarded options. For the initial random weights assigned to the neurons, we tested the following functions to generate the initial weights: Normal, Glorot Uniform, Lecun Uniform and He Normal [32]. For the neuron activation function we tested: Tanh and Rectified Linear Unit (ReLU) [33]. The grid search trained a series of networks with every possible combination of these functions and records the combination which resulted in the highest model accuracy. We omit the full grid search results for the sake of brevity but the network with the highest classification accuracy featured ReLU activation and initialisation via Glorot Uniform.

The ReLU function activates a neuron via $f(x) = \max(0, x)$, where $x$ is the incoming signal to the neuron, which thresholds the activation to stop it going below zero and is designed to more accurately imitate biological activations [33]. ReLU has been shown to improve accuracy in many Deep ANN's, whilst also improving training times. Before the weights in an ANN are updated via back-propagation, they must be assigned some random value. This initial value has been shown to have a large impact on the overall network quality [34]. The Glorot Uniform initialisation method sets the initial value for a neuron to be sampled randomly from a uniform distribution and has been shown to improve accuracy [34]. For the loss function of the network we use categorical cross-entropy, commonly used for multi-class classification tasks [35]. We used the RMSprop algorithm to update the model weights via back propagation [36]. Finally, we have included small amounts of dropout on each hidden layer, this is a regularisation strategy for ANNs which functions by randomly dropping neurons in an effort to prevent over-fitting [37]. An overview of the complete network, describing the size of each layer, the initialisation and activation functions used and the application of any dropout, is given in Table II.

Table II
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 | $\lvert \mathcal{C} \rvert$ | NA | Softmax | NA |
| Binary-Output | 1 | NA | Sigmoid | NA |

To ensure that we can also classify binary datasets (datasets for which only two unique labels are present) as well as multi-class, we created a second version of the *DTC* network. This binary class version employs an alternative output layer with a single output neuron activated via a

Sigmoid function, commonly used for binary classification tasks [38]. In addition, this network used binary cross-entropy for the loss function [35]. The alternative output layer can be seen in Table II.

### C. Implementation

The code for the *DTC* approach has been written in Python. The feature extraction code has been implemented using Graph-Tool [39]. The ANNs have been created using the TensorFlow and Keras [40] libraries, allowing us to exploit Graphics Processing Unit (GPGPU) cards to decrease training times. The SVM models have been implemented using SciKit-Learn [41].

## V. Experimentation

### A. Dataset Generation

Due to the lack of large quantities of publicly available empirical graph datasets, we created two balanced datasets synthetically using a combination of five mathematically understood random graph generation methods from the SNAP graph library [42]. The two datasets are detailed below:

**Dataset One (Multi-Class)** - Containing 10,000 graphs from each of the five generation methods, creating a final dataset of 50,000 graphs, with five balanced classes. This dataset was created to test the ability of the *DTC* approach at multi-class classification.

**Dataset Two (Binary Classification)** - Containing 10,000 forest fire graphs and 10,000 randomly rewired forest fire graphs. The goal of this dataset was to test the sensitivity of the *DTC* approach at classifying graphs which are highly topologically similar but of two different classes. The forest fire graphs represent a normal distribution of graphs, whereas the rewired graphs represent anomalies where small changes have been made to their topologies. The random rewire process modifies a given source graph's topology by randomly altering the source and target of a set number of edges according to the Erdős-Rényi random model. The number of edges each graph was rewired by was chosen uniformly from a possible range of 100 to 10,000.

Many of the graph generation methods used require parameters to control aspects of the generation process. To avoid our models over fitting to a particular set of generation parameters, these we uniformly randomised by the amounts detailed below. Each graph was generated with 100,000 vertices and a varying number of edges controlled via the generation method. The methods used were:

*Forest Fire (FF)* [43] - The forward and backward burn probabilities were chosen uniformly between 0 and 0.5.

*Barabási-Albert (BA)* [44] - The number of connections made by each new vertex joining the graph was chosen uniformly between 2 and 6.

*Erdős-Rényi* [45] - No parameters were randomised for this method as edges are made at random.

*Small World (SW)* [29] - The rewire probability for the small world model was chosen uniformly between 0 and 0.5.

*R-MAT (RM)* [46] - The R-MAT graph generator requires the probability that a certain edge will fit into one of three partitions within a $2 \times 2$ matrix. These probabilities are uniformly chosen to sum to less than one.

### B. Testing Methodology and Environment

All the accuracy scores presented in the results section are the mean accuracy after $k$-fold cross validation, considered the gold standard for model testing [47]. For $k$-fold cross validation, the original dataset is partitioned into $k$ equally sized partitions. $k-1$ partitions are used to train the model, with the remaining partition used for testing. The process is repeated $k$ times using a unique partition for testing and a mean taken to produce the final result.

Experimentation was performed on a compute system with 2 Nvidia Tesla K40c's, 20C 2.3GHz Intel Xeon E5-2650 v3, 64GB RAM and the following software stack: CentOS 7.2, GCC 4.8.5, CUDA 7.5, CuDNN v4, TensorFlow 0.10.0, Keras 1.0.8, scikit-learn 0.17.1, Boost 1.56, Python 2.7.5 and Graph-Tool 2.8.

## VI. Results

The applicability of *DTC* was tested with the two datasets described in Section V-A. Previous work has shown global graph features classified via an SVM to be more accurate than state of the art graph kernel methods [1] [6]. As the source code of the SVM approach is not publicly available, we compare *DTC* with an SVM trained upon the global features detailed in Section IV-A. Additionally we compare with an SVM trained on the full feature vector to directly assess the suitability of ANNs for graph classification.

For both the multi-class and binary classification results, six different methods are compared: *DTC-Scaled* (trained on scaled features), *DTC-Unscaled* (trained on unscaled features), *SVM-Scaled* (Trained on scaled features), *SVM-Unscaled* (Trained on unscaled features), *SVM-Global-Scaled* (Trained on scaled global features) and *SVM-Global-Unscaled* (Trained on unscaled global features).

### A. Multi-Class Classification

To assess the accuracy of the *DTC* approach at performing multi-class classification, Dataset One was used. The reported results, displayed in Table III, are the mean accuracy as a percentage of the $k$-fold cross validation along with the 95% confidence interval. The table shows that the *DTC* approach has a very high accuracy across the $k$-fold cross validation run and is over 10% more accurate than the best SVM approach. It can also be seen that using the full feature vector with the SVM is much more accurate than global features alone. The table also highlights how beneficial feature scaling is to the overall accuracy of both

Table III
MULTI-CLASS CLASSIFICATION RESULTS

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

Table IV
BINARY CLASSIFICATION RESULTS

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

models. figures 1 and 2 show the error matrices for the *SVM-Scaled* and the *DTC-Scaled* methods respectively. These figures show the predicated against the true labels. The *SVM-Scaled* approach has difficulty correctly classifying the ER, FF and SW classes, with the ER class more frequently being classified as BA than it's true class. Whereas, Figure 2 shows that *DTC-Scaled* is consistently accurate across all classes.

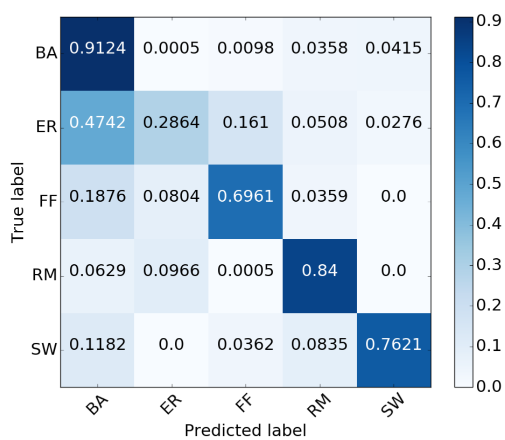



Figure 1. Normalized Error Matrix For SVM (Scaled)

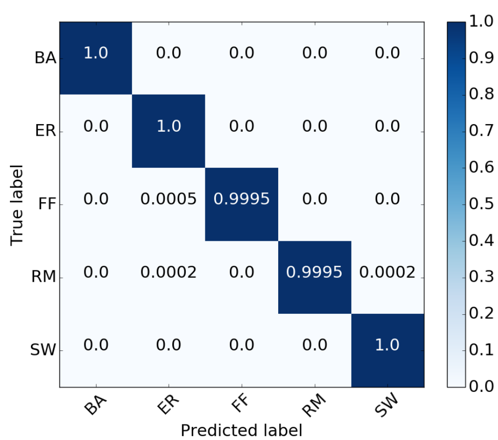



Figure 2. Normalized Error Matrix For DTC (Scaled)

### B. Binary Classification

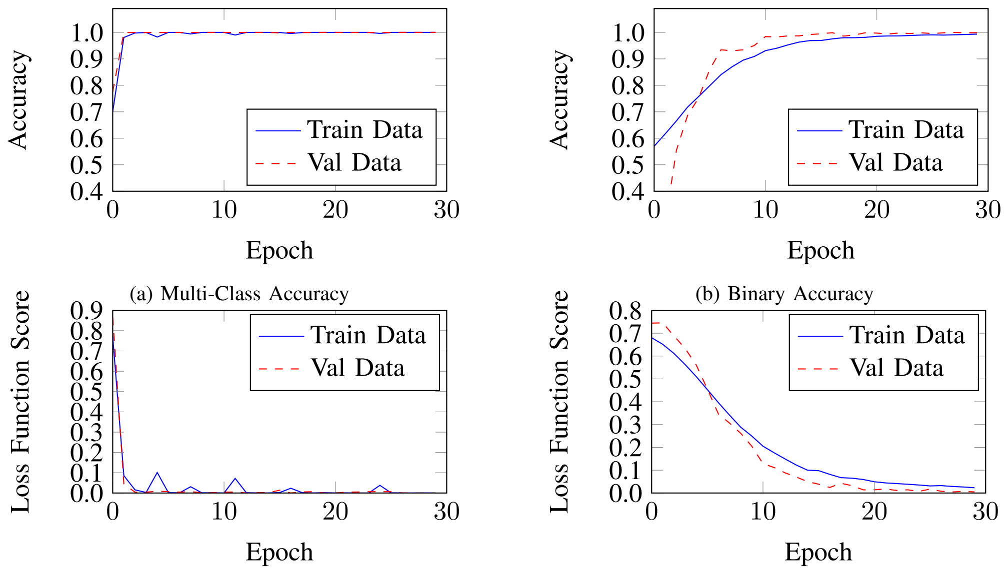
To assess the accuracy of the *DTC* approach at performing binary classification, Dataset Two was used. Here we assess the sensitivity of *DTC* when classifying graphs which are highly topologically similar. Table IV shows the results for the binary classification. *DTC* achieves a very high accuracy when detecting binary classes, with the *DTC-Scaled* approach beating the best SVM approach by over 30%. The accuracy of the *DTC* in this dataset is extremely encouraging, as the topological distribution of the two classes represented in this dataset are very close.

### C. Model Accuracy and Score Over Epochs

When training ANNs, the complete set of training data is passed through the network multiple times to further improve the accuracy, with one epoch being a complete pass through the training data. Figure 3 shows how the accuracy and loss function score of the binary and multi-class versions of the *DTC* approach respond as the number of epochs in the train and validation data are increased. Plotting the scores for the training and validation datasets is a useful indication of whether the model is over-fitting to the training data. The figure shows that both ANNs can achieve excellent accuracy with only 30 epochs. It is interesting to note that the binary network takes over 10 epochs longer to reach its maximum accuracy, demonstrating that this classification task was more complicated for the ANN to learn correctly due to the closeness of the classes. The figure also highlights that neither of the models are over-fitting to the training data, as both test and validation sets exhibit highly similar variations in accuracy and loss score.

## VII. CONCLUSION

This paper has presented a novel approach for global graph classification entitled *Deep Topology Classification*. The presented results show that the combination of extracting deep topological and global features from a graph and classifying these via a deep neural network is an




(a) Multi-Class Accuracy (b) Binary Accuracy

(c) Multi-Class Loss (d) Binary Loss

Figure 3. Model Accuracy and Loss Score Over Epochs

extremely effective approach to the problem of global graph classification. The approach is shown to have over 99% classification accuracy after $k$-fold cross validation across a multi-class and binary dataset. This compares very favourably with the current state of the art approach which has an accuracy of 88.4% for the multi-class and 68% for the binary datasets. We believe that the classification accuracy offered by the *DTC* approach will have many applications within the field of graph mining.

In future research we hope to further explore classification accuracy of the *DTC* approach upon empirical graph datasets. To achieve this we will explore the use of data augmentation techniques to allow for model training upon limited amounts of input data. We plan to create an unsupervised version of *DTC* to classify graphs with no class labels presented. Lastly, we aim to investigate graph embedding approaches to see if the features required for classification can be learned from graphs automatically.

## Acknowledgment

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research. Additionally we thank the Engineering and Physical Sciences Research Council (EPSRC) for funding.

## References

[1] G. Li, M. Semerci, and B. Yener, "Graph Classification via Topological and Label Attributes," *MLG*, 2011.

[2] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: online learning of social representations," *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

[3] G. Malewicz, M. Austern, and A. Bik, "Pregel: a system for large-scale graph processing," *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pp. 135–146, 2010.

[4] N. Doekemeijer and A. L. Varbanescu, "A Survey of Parallel Graph Processing Frameworks," 2014.

[5] S. V. N. Vishwanathan, N. N. Schraudolph, I. R. Kondor, and K. M. Borgwardt, "Graph Kernels," *Journal of Machine Learning Research*, vol. 9, pp. 1201–1242, 2010.

[6] G. Li, M. Semerci, B. Yener, and M. J. Zaki, "Effective graph classification based on topological and label attributes," *Statistical Analysis and Data Mining*, vol. 5, no. 4, pp. 265–283, aug 2012.

[7] A. Grover and J. Leskovec, "node2vec : Scalable Feature Learning for Networks," *KDD*, 2016.

[8] T. Mikolov, G. Corrado, K. Chen, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, pp. 1–12, 2013.

[9] D. Guthrie, B. Allison, and W. Liu, "A closer look at skip-gram modelling," *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*, pp. 1222–1225, 2006.

[10] L. Tang and H. Liu, "Leveraging social media networks for classification," *Data Mining and Knowledge Discovery*, vol. 23, no. 3, pp. 447–478, 2011.

[11] ——, "Scalable learning of collective behavior based on sparse social dimensions," *Proceeding of the 18th ACM conference on Information and knowledge management CIKM 09*, p. 1107, 2009.

[12] T. Gartner, P. Flach, and S. Wrobel, "On Graph Kernels: Hardness Results and Efficient Alternatives," *Proceedings of the 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop*, pp. 129–143, 2003.

[13] K. M. Borgwardt and H. P. Kriegel, "Shortest-path kernels on graphs," in *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2005, pp. 74–81.

[14] M. Thoma, H. Cheng, A. Gretton, J. Han, H. P. Kriegel, A. Smola, L. Song, P. S. Yu, X. Yan, and K. M. Borgwardt, "Discriminative frequent subgraph mining with optimality guarantees," *Statistical Analysis and Data Mining*, vol. 3, no. 5, pp. 302–318, 2010.

[15] T. Guo and X. Zhu, "Understanding the roles of sub-graph features for graph classification," *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13*, pp. 817–822, 2013.

[16] N. S. Ketkar, L. B. Holder, and D. J. Cook, "Empirical comparison of graph classification algorithms," *2009 IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2009 - Proceedings*, pp. 259–266, 2009.

[17] S. Pan and X. Zhu, "Graph classification with imbalanced class distributions and noise," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 1586–1592, 2013.

[18] "GMLC: A multi-label feature selection framework for graph classification," *Knowledge and Information Systems*, vol. 31, no. 2, pp. 281–305, 2012.

[19] Y. Yao and L. Holder, "Scalable SVM-Based Classification in Dynamic Graphs," *2014 IEEE International Conference on Data Mining*, pp. 650–659, 2014.

[20] N. Shervashidze, P. Schweitzer, V. Leeuwen, E. Jan, K. Mehlhorn, and K. Borgwardt, "Weisfeiler-Lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, pp. 2539–2561, 2011.

[21] S. Bonner, J. Brennan, I. Kureshi, G. Theodoropoulos, and A. S. McGough, "Efficient Comparison of Massive Graphs Through The Use Of Graph Fingerprints," in *Twelfth Workshop on Mining and Learning with Graphs (MLG) Workshop at KDD'16*. ACM, 2016.

[22] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 626–688, may 2015.

[23] C. Aggarwal and K. Subbian, "Evolutionary network analysis: A survey," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, pp. 1–36, 2014.

[24] B. Kantarci and V. Labatut, "Classification of Complex Networks Based on Topological Properties," in *2013 International Conference on Cloud and Green Computing*. IEEE, sep 2013, pp. 297–304.

[25] P. Bonacich, "Some unique properties of eigenvector centrality," *Social Networks*, vol. 29, no. 4, pp. 555–564, 2007.

[26] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking:bringing order to the web." 1998.

[27] M. Han, K. Daudjee, K. Ammar, M. T. Ozsu, X. Wang, and T. Jin, "An Experimental Comparison of Pregel-like Graph Processing Systems," *Vldbe*, vol. 7, no. 12, pp. 1047–1058, 2014.

[28] M. Berlingerio, D. Koutra, T. Eliassi-Rad, and C. Faloutsos, "NetSimile: A scalable approach to size-independent network similarity," *arXiv preprint*, pp. 1–8, 2012.

[29] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks." *Nature*, vol. 393, no. 6684, pp. 440–2, 1998.

[30] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 1, pp. 436–444, 2015.

[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[32] F. Chollet, "keras," https://github.com/fchollet/keras, 2015.

[33] K. Hara, D. Saito, and H. Shouno, "Analysis of function of rectified linear unit used in deep learning," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2015-September, 2015.

[34] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9, pp. 249–256, 2010.

[35] P. Golik, P. Doetsch, and H. Ney, "Cross-entropy vs. Squared error training: A theoretical and experimental comparison," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2013, pp. 1756–1760.

[36] T. Tieleman and G. Hinton, "rmsprop: Divide the gradient by a running average of its recent magnitude." *Neural Networks for Machine Learning*, 2012.

[37] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout : A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research (JMLR)*, vol. 15, pp. 1929–1958, 2014.

[38] G. Daqi and J. Yan, "Classification methodologies of multilayer perceptrons with sigmoid activation functions," *Pattern Recognition*, vol. 38, no. 10, pp. 1469–1482, 2005.

[39] T. de Paula Peixoto, "graph-tool: An efficient python module for manipulation and statistical analysis of graphs. ," 2016. [Online]. Available: https://graph-tool.skewed.de/

[40] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Man, R. Monga, S. Moore, D. Murray, J. Shlens, B. Steiner, I. Sutskever, P. Tucker, V. Vanhoucke, V. Vasudevan, O. Vinyals, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv:1603.04467v2*, p. 19, 2015.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[42] J. Leskovec and R. Sosič, "Snap: A general-purpose network analysis and graph-mining library," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 1, p. 1, 2016.

[43] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph Evolution: Densification and Shrinking Diameters," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, 2007.

[44] R. Albert and A. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, no. 1, pp. 47–97, 2002.

[45] P. Erdős and A. Rényi, "On random graphs I," *Publicationes Mathematicae*, 1959.

[46] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-MAT: A Recursive Model for Graph Mining," *Proceedings of the 2004 SIAM International Conference on Data Mining*, pp. 442–446, 2004.

[47] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection *," *Statistics Surveys*, 2010.