Proceedings of the 16th Conference on Computer Science and Intelligence Systems pp. 81–90

DOI: 10.15439/2021F140
ISSN 2300-5963 ACSIS, Vol. 25

Rotation Variance in Graph Convolutional Networks

Nguyen Anh Mac
American School of Warsaw
Warszawska 202, 05-520 Bielawa, Poland
Email: 22mac_a@aswarsaw.org

Hung Son Nguyen
University of Warsaw
Krakowskie Przedmieście 26/28, 00-927 Warszawa, Poland
Email: son@mimuw.edu.pl

***Abstract*—Convolution filters in deep convolutional networks display rotation variant behavior. While learned invariant behavior can be partially achieved, this paper shows that current methods of utilizing rotation variant features can be improved by proposing a grid-based graph convolutional network. By performing spectral graph convolutions on features extracted from subareas of images, we are able to take advantage of the geometric nature of relational machine learning in graph neural networks to be able to overcome rotation variant features to perform object localization. We demonstrate that Grid-GCN heavily outperforms existing models on rotated images, and through a set of ablation studies, we show how the performance of Grid-GCN implies that there exist more performant methods to utilize fundamentally rotation variant features and we conclude that the inherit nature of spectral graph convolutions is able to learn invariant behavior.**

I. Introduction

OBJECT LOCALIZATION, i.e., specifying an object's location within an image, is an evolving subtask of computational vision problems that have grown in prevalence in recent years.

While the usage of deep convolutional networks in object localization shows near-human level accuracy [1], [2], convolutional neural networks exhibit fundamental flaws, primarily rotation variant behavior. This lack of rotation invariance is a cost of the translation invariance present within convolutional networks. Previous works in creating rotation invariant models utilize explicit methods in order to encourage or establish learned invariance within filters [3], [4]; however, it is clear that convolutional models do not naturally learn or exhibit rotation invariant performance.

Recent advancements in Graph Neural Networks allow us to utilize Graph Neural Networks (GNNs) in performing this object localization task with rotational invariant behavior. Relational machine learning presents an interesting methodology to approach these flaws of convolutional networks by leveraging the relational nature of the rest of the connected graph. In theory, this spectral characteristic of relational machine learning permits us to establish rotational invariance in an object localization system by utilizing message passing and neighborhood aggregation. There are previous works towards object localization with relational machine learning solutions [5], [6], [7], utilizing the geometric nature of graphs to represent images. In this paper, we present a system to perform grid-based object localization using Graph Neural Networks, titled Grid-GCN. Figure 1 demonstrates the usage and output of Grid-GCN. By employing spectral graph convolutions and pre-established feature extractors, we can effectively represent images geometrically and utilize neighborhood aggregation to learn rotational invariance. We show that our model performs comparably to state-of-the-art models (namely YOLOv4 and ResNest in grid localization) in general operation and outperforms these models on rotated images, despite lacking explicit methodology to counteract the impact of rotated images. Thus, we show that relational machine learning demonstrates the ability to learn invariant behaviour that deep convolutional networks are unable to, highlighting the current ineffective use of rotation variant features.

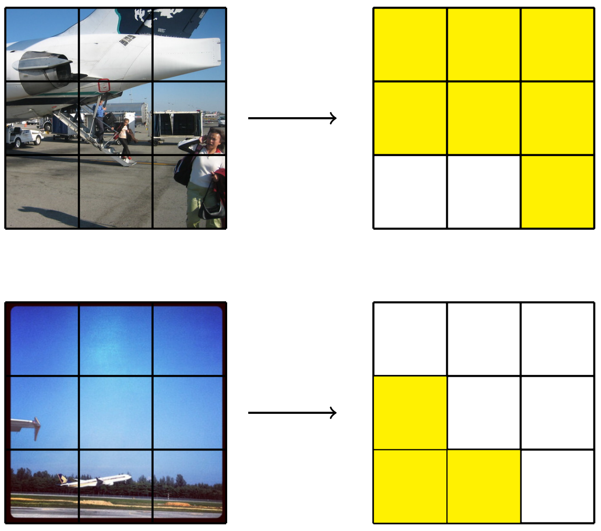


Fig. 1: **Output of Grid-GCN**. Grid-GCN outputs a grid containing the confidence that an object is present within the image. Yellow squares represent subareas where the confidence is above the detection threshold.

The structure of the manuscript is given as follows. In Section 2, we describe the previous works done regarding feature extraction, graph neural networks, and object localization. Section 3 outlines the principle concepts behind Grid-GCN and the theoretical and practical considerations of implementation.

In Section 4, we describe our experimental approach and the different datasets used to validate our approach, and Section 5 compares the results of Grid-GCN compared to existing object localization solutions. More importantly, in this section we

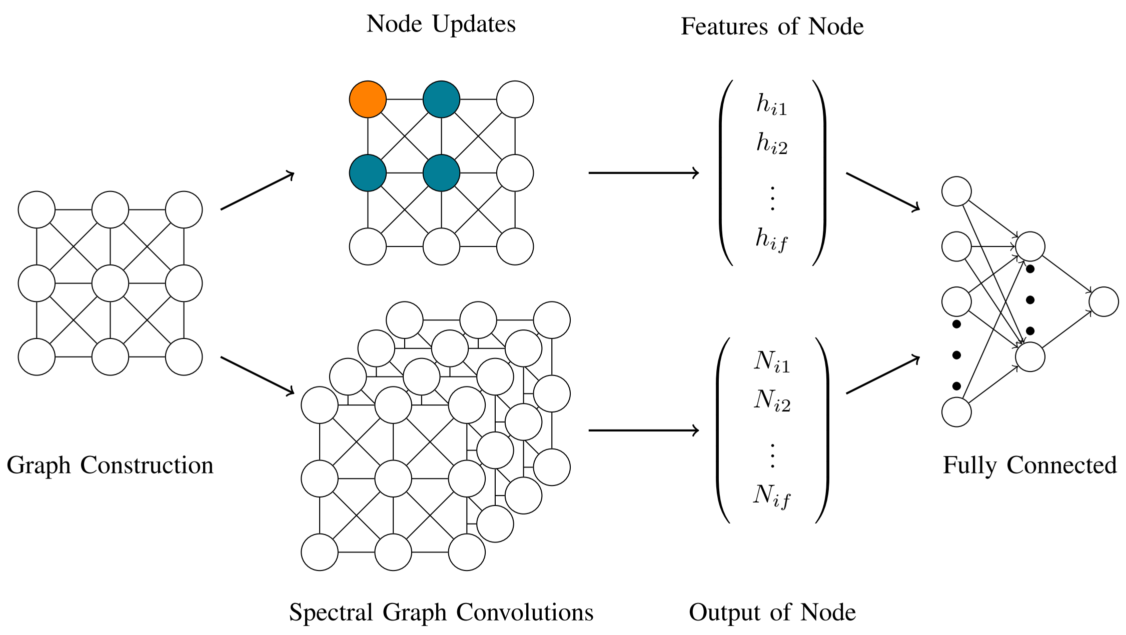

Node Updates
Features of Node
h_{i1} h_{i2} $\vdots$ h_{if}
Graph Construction
Fully Connected
N_{i1} N_{i2} $\vdots$ N_{if}
Spectral Graph Convolutions
Output of Node

Fig. 2: **Grid-GCN process.** The figure depicts the process of Grid-GCN and Graph Neural Networks. Given a graph (constructed from an image, as described in 3.1 Graph Construction), inference begins with graph convolutions, as described in 3.2 Spectral Graph Convolutions for Grid-GCN. Graph convolutions produce an output vector for each node, inserted as the input for a series of fully connected layers (equivalent to classification layers in traditional neural networks). Standard Graph Neural Networks commence with node updating, where the orange node is updated with the features of the nodes connected (represented by blue nodes). For both processes, each node produces an output vector that represents the confidence that a node contains an object.

discuss the rotation variant nature of features from convolution filters and how Grid-GCN demonstrates that there exists better methods to use features to exhibit learned invariant behaviors. In Section 6, we conclude with further discussion of the practical and future works regarding this paper's findings.

II. Previous Work

A. Feature Extraction in Deep Learning

Feature extraction is the process of representing an initial set of information into a set of informative and non-redundant values called features. We primarily focus on feature extraction in deep learning models. Most, if not all, models relating to computer vision utilize a form of feature extraction [8], [1], [2], [9], [10], [11]. By learning on classifications of a specified object, the feature extraction model can effectively learn the necessary embeddings to represent specified objects' defining features. This paper employs the usage of a pre-trained ResNet model, which is detailed in [12]. ResNet has proven to be a reliable feature extractor for a multitude of domains [12], [13], and has shown the ability to operate at a range of resolutions [14], [15], [16], [17], which is crucial for the purposes of generalized datasets such as the Common Objects in Context dataset (abbreviated to COCO) [18].

B. Object Localization

Object localization refers to locating and indicating the location of objects within an image. There have been vast improvements in object localization in recent years due to significant technical improvements and the introduction of widely available and high-quality data sets such as the Common Objects in Context (COCO) dataset [18]. In this paper, we discuss object localization from the perspective of object detection and semantic segmentation. Historically, attempts at object localization in object detection tasks evaluate region proposals generated by heuristic algorithms [19]; however, region proposal networks and embedded-region detection methods have been steadily gaining prevalence [20], [10], [11]. Popular models, such as single-shot detectors [11], utilize an embedding region proposal network to perform localization, while models such as You-Only-Look-Once (YOLO) [21] utilize embedded region information generated from feature extraction to perform localization. Object localization, from the perspective of semantic segmentation, approaches localizing by indicating the presence of an object on a pixel-wise basis. Segmentation models [22], [23], [24] often build from convolutional networks to perform pixel-wise class predictions, essentially constructing a detailed localization of an image.

C. Graph Neural Networks (GNNs)

Relational machine learning, precisely graph neural networks, are becoming increasingly prevalent for solving computational vision tasks [5], [6], [7]. Solutions with relational machine learning often utilize the geometric nature of graphs to define a unique representation of images in order to effectively learn node relationships or graph classifications [25]. In this paper, we primarily focus on node classification, formally represented as learning the label of each node using the state of each node feature-vector to be as close as the ground truth of the node.

$$y_i = o(h_i, x_i) \tag{1}$$

where y_i is the output of a node, o is the output function, h_i is a node's state embedding, and x_i is the features of node i.

Modern GNNs follow a message propagation or neighbourhood aggregation principle, in which each embedding of each node updates depending on it's neighbors [25]. GNNs can be loosely categorized into a spectral filter and spatial filter models, both of which attempt to generalize convolution into a geometric manner, thus are dubbed 'Graph Convolutional Networks' (GCNs) [26]. However, while spatial approaches to relational machine learning exist in theory, it faces challenges regarding the representations of local neighborhoods [27], [28]. The key difference between spatial and spectral approaches in spatial approaches emphasizes edges in a node's nearest K-neighbours, while spectral approaches generalize across all neighbors effectively. The authors of [29] propose spectral graph convolutions in which filters are multiplied by graph signals and processed through graph coarsening to produce accurate representations of local neighborhoods. We employ the graph convolutional process given in [29] which is further explained in 3.2 Spectral Graph Convolutions, for the proposed solution.

III. Grid-GCN

Grid-GCN is inspired by the graph convolutional network model framework outlined in [26]. While we borrow ideas from [26], the organization and applications of the GCN method is completely novel. Traditional GNNs operate on the assumption that connected nodes are likely to share the same label. This assumption, however, while not necessarily incorrect, hinders modeling capacity. As such, GCNs removes this limitation by encoding a graph structure using a neural network and training on a supervised target. The readers are referred to [26] for more details.

The process outlined in this section consists of three stages: graph construction, feature extraction, and classification. In the case of this experiment, we view an image as a ten by ten grid. Grid-GCN uses a ten by ten grid due to the size constraints of the COCO dataset [18]. In general, it is vital to consider the amount of detail present within a subarea of a grid when deciding upon the resolution of the grid in order to convey succinct features relating to the object. The initial states of each subarea of this grid are set equal to the features of this subarea, and the input for each continuous iteration of internal inference is the initial states of each subarea.

Each node updates it's hidden state for a certain amount of steps (200 in our experiment). The output of each node are then placed into a traditional classification network, which returns a confidence measure. Figure 2 illustrates the outlined process.

A. Graph Construction

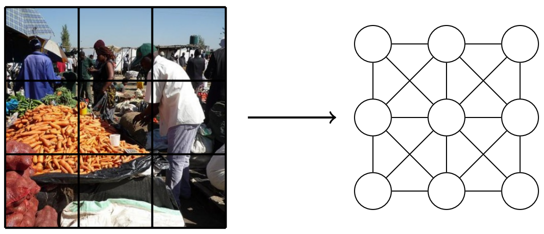


Fig. 3: **Graph Construction**. Images can be seen as a combination of subareas, which represent nodes of a graph. By doing so, we can represent an image geometrically effectively. Each node's initial state is the features of the subarea, and the weight of each connection is the cosine distance between respective initial states.

It is possible to view each image as a graph, where the nodes in the graph represent a subarea of a grid. The initial state of each node in this graph is set equal to the feature-vector extracted from the respective subarea. Figure 3 illustrates this concept. Each node is arbitrarily connected to its immediate vertical, horizontal, and diagonal neighbors on the grid structure whose edge weight is set equal to the cosine distance between the states of their respective nodes. By setting the edge weight equal to the cosine distance of the state of a node's immediate neighbor, the graph can comprehend the notion of scale and similarity. Nodes whose states are drastically different represent distinctions from background and foreground from one another; as such, their states should minimally impact each other. Formally, an edge between two nodes is determined as follows:

$$W_{ij} = \frac{s_i \cdot s_j}{||s_i||||s_j||} \tag{2}$$

where W_{ij} is the weight between nodes i and j, s_n is the state of the node n. It is important to note that since the graph is undirected, $W_{ij} = W_{ji}$.

By setting the edge of two nodes as the weight of the cosine distance of their states, objects are scaled-down, i.e., instances of an object whose size is equivalent to one subarea in the grid negatively impact neighboring nodes due to differing states. Likewise, nodes whose states are similar to one another reinforce the states of one another. By doing so, images whose objects span multiple subareas more directly influence one another.

B. Spectral Graph Convolutions

In this context, spectral convolutions offer a method to filter repeated information from a node's neighbors effectively. A node's state is not self-reinforced by an aggregate of the neighborhood with similar states but instead reinforced by a filtered state of the neighborhood, leading to diverse states during both message passing and classification. Diverse states preserve a node's original state while incorporating crucial information from the neighborhood. Mathematically, to perform effective convolutional operations on a graph, we must be able to effectively multiply a signal with a diagonalized filter on the Fourier basis. Convolution theory states that convolution in the spatial domain is equivalent to multiplication in the Fourier domain; thus, it is equivalent to eigendecomposition applied on the graph Laplacian.

$$F(x,\theta) = g_\theta(L)(x) = U g_\theta U^T x \tag{3}$$

where L is the normalized graph Laplacian $L = I_N - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ (where I_N is the identity matrix, D is the diagonal degree matrix, and W is the weighted adjacency matrix), where $g_\theta(N) = diag(\theta)$ (where $\theta \, \varepsilon \, \mathbb{R}^N$ is a vector of Fourier coefficients), and U is the matrix of eigenvectors of the normalized graph Laplacian, and x is the state of the node [29], [26].

The naive approach to this process is generally considered too inefficient to be utilized in practical cases; as such, [29] proposed an efficient implementation of the above formula by parameterizing the filter with the normalized Laplacian graph and approximating the aforementioned function as a vector whose terms are a part of the Chebyshev polynomial.

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\overline{\Lambda}) \tag{4}$$

where $T_k(x)$ is the Chebyshev polynomial $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ and $T_0 = 1$ and $T_1 = x$, and $\overline{\Lambda} = 2\Lambda{\lambda_{max}}^{-1} - I_n$ where λ_{max} represents the largest eigenvalue in L. This approximation reduces the computational complexity from $O(n^2)$ to $O(|\epsilon|)$ where ϵ is the set of edges. The reader is advised to [29] for a more detailed justification.

Consider the general equation for the features of a particular node at any given timestep t:

$$h_i^t = f(x_i^{t-1}, \bigcup_{\forall j: i \Rightarrow j} q(x_j^{t-1}, h_j^{t-1}, E_{ij})) \tag{5}$$

where f is the update function, q is the message preparation function, and E_{ij} are the features between nodes i and j. Graph Convolutional Networks specify the updating of the hidden states as follows:

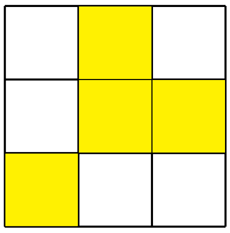
$$h_i^t = \sum_{\forall j: i \Rightarrow j} N_j^{t-1} \theta_j \tag{6}$$

where N_j is the output of a node from the convolution of the hidden state with a filter, approximated using the Chebyshev approximation, as defined as:

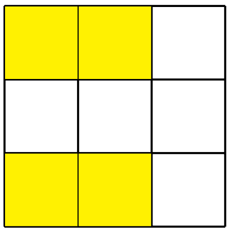
$$N_i^t = \sum_{k=0}^{K-1} \theta_k T_k(\overline{\Lambda}) \tag{7}$$

IV. The Proposed Method

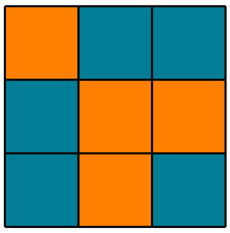
It is important to note that grid localization, as outlined in the paper, is a novel task. Its nature is inherently difficult to correctly evaluate the success of grid-based localization due to a lack of previous work dedicated to the subject. As such, we define our own measures of success. We classify a subarea as predicted "correctly" (confidence score above the detection threshold) if an object or lack thereof matches the ground truth. Figure 4 illustrates an example of subareas of predicted grids are classified as a true positive, true negative, false positive, or false negative.



(a) Predicted Grid



(b) Ground Truth Grid



(c) Samples Grid

Metric	Amount
TP	2
TN	3
FP	2
FN	2

(d) Confusion Matrix

Fig. 4: **Comparing a predicted grid to the ground truth**. In the Predicted Grid and Ground Truth Grid, yellow boxes represent subareas with an (true or predicted) object. Thus, blue boxes within the Samples Grid represent true results (true positive and true negative), while orange boxes represents false results. True positive or true negative results occur when both the predicted and ground truth match and is positive or negative respectively. Similarly, false negative results occur when the predicted and ground truth subareas mismatch, and the ground truth subarea is positive. False positive results occur when the areas are mismatched, and the ground truth area is negative.

A. Measures of Evaluation

F-Score. F-Score is a classic metric used in many other works of object detection [30]. Classically, the F-Score measures the classification accuracy in terms of the harmonic mean of the precision and recall.

$$\begin{aligned} F\text{-}Score &= 2\frac{precision \cdot recall}{precision + recall} \\ &= \frac{2TP}{2TP + FP + FN} \end{aligned} \tag{8}$$

The harmonic mean property of the F-Score ensures that a balance of identifying positives and negatives is observed. This property is useful in scenarios with a large imbalance of positives and negatives. For example, consider the case where the rate of a positive sample to a negative sample is 1:99. A model that identifies all samples as negative returns an accuracy rate of 99%, however, is not particularly helpful in this context. Such a model, in contrast, has an F-Score of 0. In this experiment and practical applications of Grid-GCN, there is an imbalance of negative samples and positive samples (particularly skewed towards negative samples). Thus, F-Score is an appropriate measure of the success of Grid-GCN in relation to pre-existing models.

Matthews Correlation Coefficient. Matthews Correlation Coefficient (MCC) or phi coefficient, much like F-Score, is a popular statistic in machine learning [31], [32]. MCC is viewed as a balanced measure of true negative, true positive, false negative, and false positive samples. Specifically, MCC balances the performance of detecting each category of samples despite large imbalances in sample rates. MCC is defined as the following expression:

$$\frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{9}$$

The expression above produces results between -1 and 1, where -1 symbolizes the opposite correlation between prediction and truth, 0 is equivalent to random guessing, and 1 represents perfect predictions. While F-Score is an effective statistic that accurately depicts the performance of a model, there are particular imbalances in data that cause misleading results, primarily due to the fact that F-Score does not consider true negatives. For example, consider the scenario where a model detects 100 true positive results, five false positive results, one true negative, and nine false negative results. This would produce an F-Score of 93.46%, however, notice that the model is only detecting one negative sample out of a total of 6 (one true negative and five false positive samples). Thus, this model's MCC is 6.64%, reflecting the model's poor performance in detecting negative samples. Therefore, using MCC in conjunction with F-Score will accurately portray Grid-GCN's performance compared to other models.

B. Model Topology

The topology of Grid-GCN consists of three distinct, separable 'sections,' the feature extraction, graph convolution, and classification section. As mentioned previously, the feature extraction section was implemented with a pretrained Resnet-50, whose weights were frozen. The graph convolution section consists of two distinct spectral graph convolution layers. The function of these layers are detailed in 3.2 Spectral Graph Convolutions. We chose a filter size of three for this experiment, and hidden dimensions of 512 and 256 were chosen for the first and second convolution layers, respectively. A batch normalization and the RELU activation operator were applied to each of these layers. During the Grid-GCN training, a dropout of 0.5 was applied to the output of the second convolution layer. Dropout layers within GNNs have proven to be significantly important in the success of the generalization of models during training. The classification section of Grid-GCN consisted of two fully-connected layers of a hidden dimension of 256. Batch normalization and RELU were applied to the output of the first classification layer, while the activation function of the second layer was sigmoid. For each node, Grid-GCN outputs a vector representing the confidence of each class.

C. Model Configuration

For this paper, Grid-GCN is implemented using Pytorch [33] 1.7.0, Pytorch-Geometric [34] 1.6.3 and Tensorflow [35] 2.4.0 on 2 Nvidia K80 GPUs. For feature extraction, our experiment utilizes the Resnet-50 model trained on 'ImageNet' weights included in the Keras interface, and the fast spectral convolution operations described in 3.2 Spectral Graph Convolutions is implemented in Pytorch-Geometric by default. ResNet-50 uses a pooling mode of average, and we rescale the subareas in the grid to 224x224. The Adam [36] optimizer, included in Pytorch 1.7.0, whose hyperparameters were set to a learning rate of 0.01 and a weight decay of $5 \cdot 10^{-4}$ is used. Each image is processed and trained in 200 steps.

D. Dataset

We use the COCO Dataset [18] which is divided into three subsets, training, validation, and testing images, of which there are 118k images in training, 5k in validation, and 41k in testing. Each of these images is labeled using instances, of which each instance represents an object (with a predetermined class) and a mask. As such, we are able to translate the full masks in each image into a grid, where each subarea a scalar representing the presence of an object.

E. Training

Standard data augmentation procedures, consisting of random scaling, rotation, mirroring is applied. We trained the model using cross entropy loss, which has the following formula:

$$L(y, \hat{y}) = -\sum_{k}^{K} y^k \log \hat{y}^k \tag{10}$$

where K represents the number of nodes (of which there were 100).

Validation or testing labels is not used to train the model, and results mentioned in Table 1 are labels and samples exclusively from the COCO testing set.

Standard					Rotated (randomly)				
Model	Recall	Precision	F-Score	MCC	Model	Recall	Precision	F-Score	MCC
YOLOv4	0.7076	0.8398	0.7681	0.6001	YOLOv4	0.6851	0.8042	0.7399	0.5549
ResNest	0.6079	0.8486	0.7084	0.5406	ResNest	0.5171	0.7056	0.5969	0.3521
Grid-GCN	0.6941	0.8322	0.7569	0.5953	Grid-GCN	0.6871	0.8215	0.7483	0.5742

TABLE I: Recall, precision and F-Score of YOLOv4, ResNest, and Grid-GCN on the COCO dataset with and without rotation.

F. Baseline Models

YOLOV4. YOLOv4 [1] is a frontier object detection model, whose trained weights are made public[1]. Recently, YOLOv4 achieved an mean average precision (abbreviated to mAP) of 43.5% [1] on the COCO [18] dataset, which is the model that we utilized. While a newer version of the YOLO series (YOLOv5) has been made available, a technical report regarding the construction and improvements of YOLOv5 has not been made public, hence why YOLOv4 is used instead. Given that YOLOv4 is an object detection model which produces bounding boxes, several adjustments must be made. To compare these two models, we consider each subarea whose corners are within the predicted bounding box to be a positive result. Figure 5 illustrates comparing bounding boxes to a grid. Naturally, object detection models are disadvantaged in comparison to Grid-GCN in terms of Intersection-over-Union (IOU) simply due to the fact that object detection models are trained to perform localization in bounding boxes. The model incorporates additional subareas into the final positive sample count. Unfortunately, this is unavoidable as bounding boxes are the predominant solution to non-segmentation object localization. We create the final prediction of each subarea by assigning each unique class's values that intersects (or is present) within the subarea.

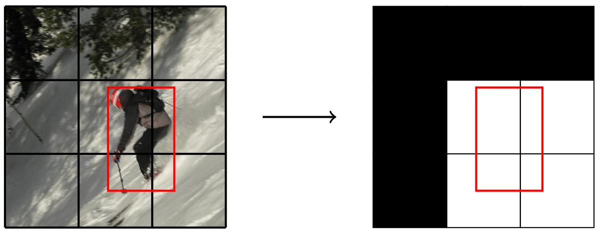


Fig. 5: **Comparing bounding boxes to a grid**, where white boxes represent positive subareas and black boxes represent negative subareas. The figure shows how object detection models are able to be compared to Grid-GCN by simply representing subareas as a collection of classes whose bounding box intersects the subarea.

ResNeSt. ResNeSt [23] is a state-of-the-art mode semantic segmentation model, achieving a mean Intersection over Union (mIoU) of 47.6% on the ADE20K validation set [23]. ResNeSt utilizes split attention networks and a unified computation block to outperform previous models in detection accuracy, classification accuracy, and computation time. At the time of writing, ResNeSt is the most accurate model (detection-wise) for the PASCAL-context dataset. For this experiment, we are utilizing ResNeSt-269, whose pretrained weights were made public [2]. Logistically, comparing the ResNeSt-269 model to Grid-GCN is trivial. We will perform a standard inference on the image (generating a mask) and divide it into an identical grid to the grid in Grid-GCN. Figure 6 illustrates a conversion from a mask to a grid.

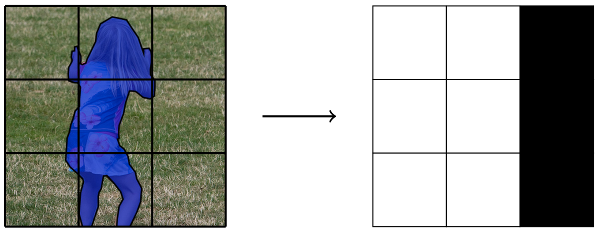


Fig. 6: **Masks on a grid**, where white boxes represent positive subareas and black boxes represent negative subareas. The figure shows how semantic segmentation models are able to be compared to Grid-GCN by simply representing subareas as positive if the mask overlaps. Image, with mask, is taken from COCO dataset's website using the explore feature[3].

V. Results

Standard operation. We define standard operation as inference and localization based on unmodified or unaugmentated images from the COCO dataset. As seen in Table 1, Grid-GCN performs comparably to YOLOv4 and significantly outperforms ResNest in all metrics. Low recall from the ResNest models suggests that the false negatives are significantly prevalent, though the poor behaviour of ResNest is difficult to explain.

Percent Decrease of Model Performance				
Model	Recall	Precision	F-Score	MCC
YOLOv4	3.18%	4.23%	3.67%	7.53%
ResNest	14.94%	16.85%	15.74%	34.87%
Grid-GCN	1.01%	1.29%	1.14%	3.54%

TABLE II: Percent decrease in performance of models where images were rotated randomly, on the COCO dataset.

Rotated Images. We perform a similar evaluation in this subcategory as the standard operation with the images rotated at random angles. YOLOv4, and approaches to computational

[1]https://github.com/pjreddie/darknet

[2]https://github.com/zhanghang1989/ResNeSt

[3]https://cocodataset.org/#explore

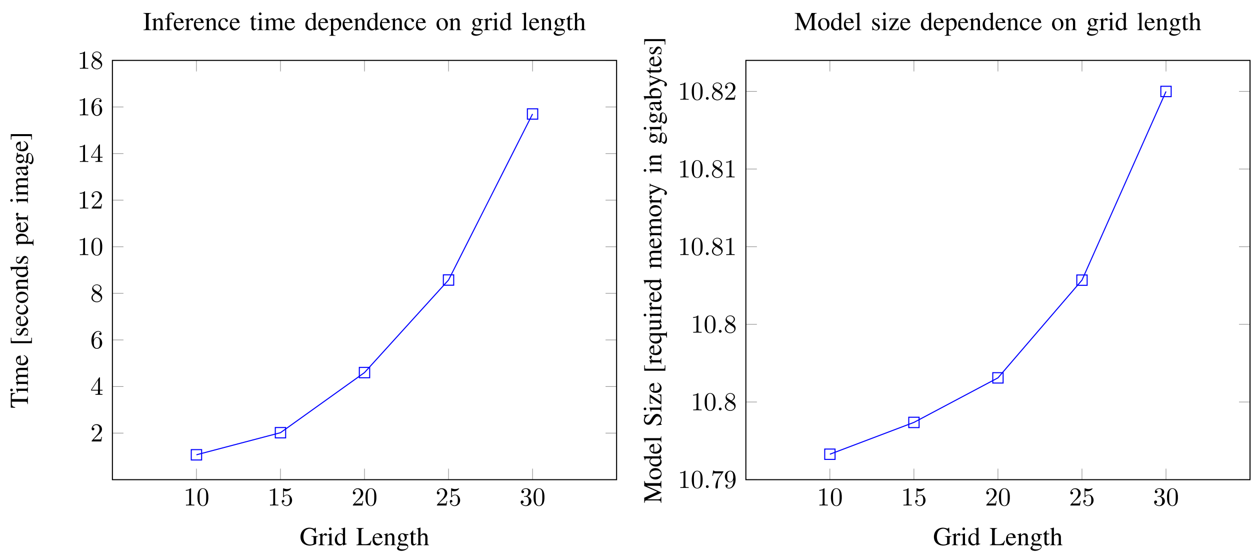


Fig. 7: **Model size and inference time dependence on grid length.** It is evident that both the size of the model (including graph information) and the inference time is exponentially correlated to the length of the grid. This exponential relationship is primarily driven by the fact that as grid size increases, the number of nodes in the graph increases exponentially.

vision tasks as a whole, rely on convolutional techniques to be able to reduce the dimensionality of data into a meaningful representation. It is known that convolutions, while being translation invariant do not exhibit rotational invariance. In more recent technology, the impact of rotational variance was reduced primarily by rotation augmentation in data, allowing filters to condition some form of rotational invariance. The impact of rotation is evident in both the YOLOv4 and ResNest models' performance after the images have been rotated, where there is a noticeable decrease in all performance metrics, as seen inTable 2. However, the diagonalization effect of bounding boxes may play a role in the decreased precision and recall for YOLOv4. Specifically, the diagonalization effect of bounding boxes increases the area of labeled positive samples, as the bounding box covers a more extensive than necessary area due to the nature of the output of such models. The diagonalization effect may explain the more pronounced decrease in precision for the YOLOv4 model. However, the drastic decrease in MCC suggests that the diagonalization effect was overshadowed by the increase in false negative results, implying that this diagonalization effect did not as heavily impact the performance of YOLOv4 as the rotational invariant nature of the model did.

While there was a decrease in recall and precision for Grid-GCN when images were rotated, this decrease was not as pronounced as the decrease for YOLOv4 and ResNest. Though this rotational invariance is not concrete (there was still a decrease in both recall and precision when images were rotated), the impact of rotation on Grid-GCN was significantly less than the impact on both YOLOv4 and ResNest, as shown in Table 2 by the lower percentage decrease in all performance metrics. It is important to emphasize that no explicit techniques to enforce rotational invariance were included within Grid-GCN. In other terms, Grid-GCN was able to learn a soft form of rotation invariance despite having rotation variant features. Previous works attempting rotation invariant image classification utilized explicit techniques [3], [4], whereas Grid-GCN learned rotation invariant behavior in an unsupervised fashion.

Grid-GCN's performance prompts the larger conversation of whether there is an aspect of features generated from convolution filters which contain a higher degree of rotation invariance than previously thought. Fundamentally, learned invariance in convolutional networks are caused by training pooling units which immediately proceed convolution filters [37]. These pooling layers allow a degree of rotation, in learned examples, to provide approximate or identical features from convolutional filters; hence demonstrating learned invariance. However, overabundance of pooling units causes loss of vital detail, thus, successful models which demonstrate learned invariance achieve a balance of pooling units to preserve both detail and invariance. According to their respective authors, all compared models and the ResNet backbone do achieve this balance [12], [1], [23]. Despite the input features of Grid-GCN being rotation variant, the comparatively minimal impact of rotation on Grid-GCN compared to both ResNeSt and YOLOv4 suggests there exists better methods to utilize learned variance in features.

To further understand the origin of Grid-GCN's learned invariance, we conducted a set of ablation studies to examine the behavior of our model. In the first ablation study, we removed the grid and feature extractor aspect of Grid-GCN, and we solely focused on if this invariant behavior originated from spectral graph convolutions. This was done by changing the size of the grid to 100 by 100, and resizing images to 100 by 100. Next, the initial states of nodes were not features extracted from ResNet, rather, normalized RGB values. After training this model under similar conditions, we compared the performance of such a model with standard and rotated images (labelled as "Non-Grid GCN" in the table below). In the second ablation study, we removed the spectral convolution aspect of the model. We still represented each image as a ten by ten grid with the initial state of each node still being output features from the ResNet backbone. Instead of using a GCN, these features were fed into a Support Vector Machine (SVM), and this SVM classified each node within the grid. After training this model under similar conditions, we compared the performance of such a model with standard and rotated images (labelled as "Grid SVM" in the table below)

Percent Decrease of Model Performance				
Model	Recall	Precision	F-Score	MCC
YOLOv4	3.18%	4.23%	3.67%	7.53%
ResNest	14.94%	16.85%	15.74%	34.87%
Grid-GCN	1.01%	1.29%	1.14%	3.54%
Non-Grid GCN	2.11%	3.02%	2.83%	5.74%
Grid SVM	5.33%	5.77%	4.14%	7.98%

TABLE III: Percent decrease in performance of models where images were rotated randomly, on the COCO dataset, including the modified GCN and Grid SVM.

In all metrics, Non-Grid GCN displayed less of a performance decrease in comparison to Grid SVM. This indicates that the impact of learned invariance from the ResNet backbone is not as impactful as the learned invariance originating from spectral graph convolutions. The exact reasons as to why this is the case is still unclear, however, a likely hypothesis is that it originates from the fact that adjacent nodes are taken into consideration during the spectral convolutions. Analogous to how convolutional networks learn invariant behavior due to pooling layers, it maybe be possible that the fact that the state of neighboring nodes are propagated act as a form of pooling. Furthermore, both of these models were more heavily impacted by rotation than Grid-GCN, suggesting that the minimal impact on the model's performance is attributed to both the rotation invariant behavior of the ResNet backbone and the dynamic nature of graph convolutions, not either/or. This implies that both the learned invariant behavior of the ResNet backbone and the consideration of neighboring nodes in spectral graph convolutions play a vital role in the learned invariant behavior of Grid-GCN.

In summary, Grid-GCN performs comparably to models in the metrics defined for this experiment (namely F-Score and MCC). Despite having no explicit rotation invariant aspects, Grid-GCN learned a soft-form of rotational invariant behavior and thus mitigated the impact of rotation variant filters on rotated images and outperformed state-of-the-art models on rotated images. It is likely that the origins of this learned invariant behavior results from a combination of the invariant behavior of the ResNet backbone and the dynamic nature of graph convolutions.

VI. Discussion

The most major limitation of Grid-GCN is the nature of grid localization. For the sake of demonstrating the fact that rotation variant features could be better utilized, Grid-GCN performs object localization on a grid (justification for such a choice is further explained in the previous section). Practically speaking, this design choice heavily limits the use cases for Grid-GCN. While there exists tasks which require grid-based object localization, namely subtasks of object detection [38], [39], [40], grid-based object localization by itself is rare. However, it is important to consider the fact that the primary goal of this paper was to emphasize the idea that current models are ineffective at utilizing rotation variant features and that graph neural networks are able to display invariant behavior. Grids are limited yet necessary drawback to use graphs in computer vision.

Another consideration for practical uses of Grid-GCN is the inference time per image. On average, Grid-GCN spends 1068 ms per image, while YOLOv4 has an inference time of 155 ms, and ResNest has an inference time of 407 ms with the implementation outlined in Section 4.3. This vast imbalance of time between compared models and Grid-GCN can be attributed to the graph construction. In the process of graph construction, feature extraction of a subarea occurs n^2, 100 in the case of this experiment, times. Moreover, the process of graph convolutions is, in its nature, slower than existing filters and standard convolutions as each image require a series of steps on processing (200 in the case of this experiment), thus, why Grid-GCN is significantly slower than the compared models. Ideally, developing a method to parallelize feature extraction across subareas (as they are independent of one another) may offer a significant improvement to inference time. Future works for accelerating graph processing on the hardware level will also offer a significant improvement of inference time [41].

Another factor is the scalability of the model. As the resolution of the grid increases, the resources (namely memory and time) required to sustain the model increase exponentially. Figure 7 illustrates both the inference time per image and the memory required to sustain the model as the grid length increases. Though the model itself does not grow exponentially (the model's memory processing is static), the graph information grows exponentially. Thus, the exponential nature of grids prohibits high-resolution versions of Grid-GCN in a practical manner both in time constraints and in-memory constraints.

VII. Conclusion

We have introduced a grid-based relational learning framework for object localization using graph convolutional networks. We show that our framework was able to display rotational invariant behavior, outperforming state-of-the-art object localization models on rotated planes despite lacking explicit methodology to enable invariant performance. Thus, we show that GCNs are able to implicitly learn invariant behaviour that deep convolutional networks are unable to. Moreover, we discuss the origins of the learned invariant behavior in Grid-GCN, namely considering spectral graph convolutions and the ResNet backbone through a set of ablation studies.

Our paper highlights two distinct topics of interest for future work. The methodology in which relational machine learning learns invariant behaviour in a more effective manner than traditional machine learning frameworks is still unclear. Exploring this methodology can be done by modifying and alternating any aspect of the Grid-GCN process. Moreover, viability of the outlined processes is dependent on alleviating time and memory constraints associated with graph machine learning.

References

[1] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[2] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 261–318, Oct. 2019. [Online]. Available: https://doi.org/10.1007/s11263-019-01247-4

[3] J. Kim, W. Jung, H. Kim, and J. Lee, "Cycnn: A rotation invariant cnn using polar mapping and cylindrical convolution layers," 2020.

[4] D. Marcos, M. Volpi, and D. Tuia, "Learning rotation invariant convolutional filters for texture classification," *2016 23rd International Conference on Pattern Recognition (ICPR)*, Dec 2016. [Online]. Available: http://dx.doi.org/10.1109/ICPR.2016.7899932

[5] W. Shi and R. Rajkumar, "Point-GNN: Graph neural network for 3d object detection in a point cloud," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2020. [Online]. Available: https://doi.org/10.1109/cvpr42600.2020.00178

[6] A. Luo, X. Li, F. Yang, Z. Jiao, H. Cheng, and S. Lyu, "Cascade graph neural networks for RGB-D salient object detection," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XII*, ser. Lecture Notes in Computer Science, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., vol. 12357. Springer, 2020, pp. 346–364. [Online]. Available: https://doi.org/10.1007/978-3-030-58610-2_21

[7] Y. Wang, K. Kitani, and X. Weng, "Joint object detection and multi-object tracking with graph neural networks," in *Proceedings of (ICRA) International Conference on Robotics and Automation*, May 2021.

[8] A. O. Salau and S. Jain, "Feature extraction: A survey of the types, techniques, applications," in *2019 International Conference on Signal Processing and Communication (ICSC)*. IEEE, Mar. 2019. [Online]. Available: https://doi.org/10.1109/icsc45622.2019.8938371

[9] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in *2014 Science and Information Conference*. IEEE, Aug. 2014. [Online]. Available: https://doi.org/10.1109/sai.2014.6918213

[10] Z. Chen, X. Jin, B. Zhao, X. Wei, and Y. Guo, "Hierarchical context embedding for region-based object detection," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXI*, ser. Lecture Notes in Computer Science, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., vol. 12366. Springer, 2020, pp. 633–648. [Online]. Available: https://doi.org/10.1007/978-3-030-58589-1_38

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2016. [Online]. Available: https://doi.org/10.1109/cvpr.2016.90

[13] D. Rukhovich, K. Sofiiuk, D. Galeev, O. Barinova, and A. Konushin, "Iterdet: Iterative scheme for object detection in crowded environments," in *Structural, Syntactic, and Statistical Pattern Recognition - Joint IAPR International Workshops, S+SSPR 2020, Padua, Italy, January 21-22, 2021, Proceedings*, ser. Lecture Notes in Computer Science, A. Torsello, L. Rossi, M. Pelillo, B. Biggio, and A. Robles-Kelly, Eds., vol. 12644. Springer, 2020, pp. 344–354. [Online]. Available: https://doi.org/10.1007/978-3-030-73973-7_33

[14] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, "Fixing the train-test resolution discrepancy," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 8250–8260. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/d03a857a23b5285736c4d55e0bb067c8-Abstract.html

[15] Z. Lu, X. Jiang, and A. Kot, "Deep coupled ResNet for low-resolution face recognition," *IEEE Signal Processing Letters*, vol. 25, no. 4, pp. 526–530, Apr. 2018. [Online]. Available: https://doi.org/10.1109/lsp.2018.2810121

[16] M. Kawulok, P. Benecki, S. Piechaczek, K. Hrynczenko, D. Kostrzewa, and J. Nalepa, "Deep learning for multiple-image super-resolution," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 6, p. 1062–1066, Jun 2020. [Online]. Available: http://dx.doi.org/10.1109/LGRS.2019.2940483

[17] A. Zhou, Y. Ma, Y. Li, X. Zhang, and P. Luo, "Towards improving generalization of deep networks via consistent normalization," *CoRR*, vol. abs/1909.00182, 2019. [Online]. Available: http://arxiv.org/abs/1909.00182

[18] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," in *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, ser. Lecture Notes in Computer Science, D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., vol. 8693. Springer, 2014, pp. 740–755. [Online]. Available: https://doi.org/10.1007/978-3-319-10602-1_48

[19] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. IEEE Computer Society, 2014, pp. 580–587. [Online]. Available: https://doi.org/10.1109/CVPR.2014.81

[20] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017. [Online]. Available: https://doi.org/10.1109/TPAMI.2016.2577031

[21] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 779–788. [Online]. Available: https://doi.org/10.1109/CVPR.2016.91

[22] A. Tao, K. Sapra, and B. Catanzaro, "Hierarchical multi-scale attention for semantic segmentation," *CoRR*, vol. abs/2005.10821, 2020. [Online]. Available: https://arxiv.org/abs/2005.10821

[23] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. J. Smola, "Resnest: Split-attention networks," *CoRR*, vol. abs/2004.08955, 2020. [Online]. Available: https://arxiv.org/abs/2004.08955

[24] Y. Yuan, X. Chen, and J. Wang, "Object-contextual representations for semantic segmentation," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VI*, ser. Lecture Notes in Computer Science, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., vol. 12351. Springer, 2020, pp. 173–190. [Online]. Available: https://doi.org/10.1007/978-3-030-58539-6_11

[25] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020. [Online]. Available: https://doi.org/10.1016/j.aiopen.2021.01.001

[26] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl

[27] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014. [Online]. Available: http://arxiv.org/abs/1312.6203

[28] Q. Liu, M. Kampffmeyer, R. Jenssen, and A. Salberg, "SCG-Net: Self-Constructing Graph Neural Networks for Semantic Segmentation," *CoRR*, vol. abs/2009.01599, 2020. [Online]. Available: https://arxiv.org/abs/2009.01599

[29] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 3837–3845. [Online]. Available: https://proceedings.neurips.cc/paper/2016/hash/04df4d434d481c5bb723be1b6df1ee65-Abstract.html

[30] D. Arya, H. Maeda, S. K. Ghosh, D. Toshniwal, H. Omata, T. Kashiyama, and Y. Sekimoto, "Global road damage detection: State-of-the-art solutions," in *IEEE International Conference on Big Data, Big Data 2020, Atlanta, GA, USA, December 10-13, 2020*, X. Wu, C. Jermaine, L. Xiong, X. Hu, O. Kotevska, S. Lu, W. Xu, S. Aluru, C. Zhai, E. Al-Masri, Z. Chen, and J. Saltz, Eds. IEEE, 2020, pp. 5533–5539. [Online]. Available: https://doi.org/10.1109/BigData50022.2020.9377790

[31] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (MCC) over f1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, Jan. 2020. [Online]. Available: https://doi.org/10.1186/s12864-019-6413-7

[32] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using matthews correlation coefficient metric," *PLOS ONE*, vol. 12, no. 6, p. e0177678, Jun. 2017. [Online]. Available: https://doi.org/10.1371/journal.pone.0177678

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[34] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *CoRR*, vol. abs/1903.02428, 2019. [Online]. Available: http://arxiv.org/abs/1903.02428

[35] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16. USA: USENIX Association, 2016, p. 265–283.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[37] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[38] J. Gao, T. Zhang, and C. Xu, "Graph convolutional tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[39] A. Nicolicioiu, I. Duta, and M. Leordeanu, "Recurrent space-time graph neural networks," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/383beaea4aa57dd8202dbff464fee3af-Paper.pdf

[40] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, "A^2-nets: Double attention networks," *CoRR*, vol. abs/1810.11579, 2018. [Online]. Available: http://arxiv.org/abs/1810.11579

[41] A. Auten, M. Tomei, and R. Kumar, "Hardware acceleration of graph neural networks," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.