DIAG2GRAPH: REPRESENTING DEEP LEARNING DIAGRAMS IN RESEARCH PAPERS AS KNOWLEDGE GRAPHS

Aditi Roy, Ioannis Akrotirianakis, Amar V. Kannan, Dmitriy Fradkin
Arquimedes Canedo, Kaushik Koneripalli, Tugba Kulahcioglu

Siemens Corporate Technology, 755 College Road East, Princeton, NJ 08540

ABSTRACT

'Which are the segmentation algorithms proposed during 2018-2019 in CVPR that have CNN architecture?' Answering this question involves identifying and analyzing the deep learning architecture diagrams from several research papers. Retrieving such information poses significant challenge as most of the existing academic search engines are primarily based on only the text content. In this paper, we introduce Diag2Graph, an end-to-end framework for parsing deep learning diagram-figures, that enables powerful search and retrieval of architectural details in research papers. Our proposed approach automatically localizes figures from research papers, classifies them, and analyses the content of the diagram-figures. The key steps in analyzing the figure content is the extraction of the different components data and finding their structural relation. Finally, the extracted components and their relations are represented in the form of a deep knowledge graph. A thorough evaluation on a real-word annotated dataset has been done to demonstrate the efficacy of our approach.

***Index Terms*—** Diagram parsing, knowledge graph, deep learning, curator, similarity analysis

1. INTRODUCTION

In the last decade, a whopping 1,710,000 research papers have been published in this area of Deep Learning (DL)[1]. Managing this large growth of DL publications has been a challenge for the researchers during validation, approval and dissemination DL related scientific information. Existing academic search engines like Google Scholar, arxiv, NOA [1] are inherently limited by their data mining and indexing approach which is restricted to only the text content of the papers.

Given that scientific advancements require conceptualization, explanations and reproducible implementations, it is important to have a uniform representation that holistically represents a scientific publication as a machine curatable model for easy dissemination of scientific facts. To address this unique requirement, we propose to create a Deep Knowledge Graph (DKG) repository for papers related to DL algorithms and methods to help improve search and retrieval of relevant information in the academic domain. A RDF knowledge graph (KG) [2] encodes semantic information as uniquely identifiable entities and relationships between them in the form of a Subject-Predicate-Object (SPO) triple. Common machine queryable representation like KG of each scientific paper will allow scientists and engineers to explore the vast knowledge that resides in these papers and identify atomic scientific facts. Taking inspiration from large-scale social KGs such as YAGO4 [3] and DBPedia5 [4] that supports various general knowledge-related queries, we develop DKG for the first time to the best of our knowledge.

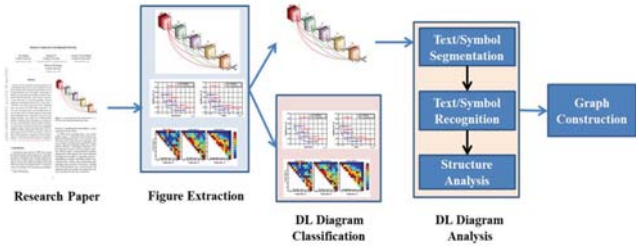

Fig. 1: Architecture of Diag2Graph to extract and represent DL diagrams in a research paper using knowledge graph.

To achieve our goal, we propose a novel framework that automatically parses a research paper to extract DL architecture related information. It has been observed that most of the research papers explain the DL model through a figure. So, in this paper we focus on representing the DL diagram figure through a DKG. Our novel end-to-end framework named Diag2Graph (see figure 1) automatically localizes all figures from a research paper, classifies them, and extracts the content of the DL diagram figures. Our proposed approach can localize a variety of figures and sub-figures, classify them by leveraging deep neural nets. As part of this work, we also introduce thorough evaluation metrics, along with a fully-annotated real-world dataset to demonstrate the efficacy of our parsing approach. Finally, to demonstrate the potential unleashed by our approach, we present a Graph AutoEncoder

This work was supported by DARPA grant HR00111990010.

[1] https://scholar.google.co.in/scholar?hl=en&as_sdt=1%2C5&as_ylo=2010&as_yhi=2019&as_vis=1&q=deep+learning&btnG=

(GAE) based KG embedding method that allows users to find papers with similar diagrams in terms of type, area and many othres. Thus, the key contributions are: (i) introduce and study the problem of DL diagram figure parsing, (ii) propose a pipeline that automatically localizes figures, classifies them, analyzes their content and represents it through a deep knowledge graph, (iii) present a thorough evaluation on a real-word dataset to demonstrate the efficacy of our approach, (iv) demonstrate the utility of DKG.

2. DIAGRAM PARSING: DIAG2GRAPH

Document analysis community has devoted much attention towards analyzing the document text content compared to figure content [5, 6]. Although scholarly figures are more structured than natural images, analysis of figures exposes a plethora of complex vision challenges due to strict requirements, high variation, heavy clutter and deformation in a figure [7]. Moreover, unlike natural image recognition tasks where desired amount of labeled training data is obtainable, figure parsing has additional challenge due to presence of only one exemplar (i.e., the legend symbol) for model learning. Although diagram recognition is a well-studied problem [8], most of the works perform analysis on flowcharts, UML diagrams, finite automata [9, 10], which follow predefined structure. Unlike these diagrams, DL architecture figures do not follow any specific structure and formatting, thus making the analysis very challenging. Our Diag2Graph framework consists of four major steps, as shown in Figure 1: (i) extracting all the figures from a research paper, (ii) segregating the figures showing DL diagram, (iii) performing content extraction from diagram, (iv) constructing DKG.

2.1. Figure Extraction from Research Paper

It is a challenging task to extract a figure as a whole as the vector images are generally embedded in the PDF document and a large figure may have multiple sub-figures. Extensive research has been done on extraction of visual figures from a PDF document [11, 12] by processing the PDF primitives. The work of [13] is interesting as it extracts a wide variety of figures along with their captions. In this paper, we build upon this work. To handle the limitation of extracting sub-figures from a figure in [13], we apply axis-aligned splits [7].

2.2. Figure Classification

The figures in DL research papers could be of different types, like, DL architecture diagrams, natural images, result figures, plots, etc. So, the first step is to segregate the DL diagrams from all the other figures extracted from a research paper. Classifying scholarly figures has recently become an area of research interest [14]. In this paper, we leverage the recent success of CNNs and present a binary neural network classifier trained on deep features extracted from fully connected layer of pre-trained CNN model. We evaluate two network architectures: VGG19 [15] and Resnet-50 [16], both pre-trained on 1.2 million images from ImageNet [17] and then fine-tuned for our figure classification task.

As the DL diagrams show extreme variations and typically do not follow any definition, it is difficult to parse them all using the same image processing framework without taking into account their variations. The taxonomy suggested in [18] is followed to train a multi-class classifier as described before to identify relevant diagrams. Among the five categories, 2D-Box plot is found to be the most popular and frequent one, thus considered for further analysis.

2.3. Content Extraction from DL Diagram

Given all the segregated DL diagrams, we next analyze their content to obtain their corresponding detailed structured representation. This involves detecting the nodes, edges connecting the nodes, and the text describing different nodes or edges.

Node Detection: After image quality enhancement with application of image pre-processing techniques, thresholding is employed to binarize the image, followed by image contour detection. However, as the diagram images often contain overlapping components, application of traditional contour detection technique on the binary images may not extract all individual nodes. We apply iterative region growing technique (which is especially useful when detecting overlapping nodes) to identify closed contours of the nodes. The contours extracted from these two techniques may have multiple bounding box detection corresponding to the same node. So, non-max suppression is applied to filter out weak bounding boxes based on their solidity indices.

Text Detection: The text description in a diagram image is obtained by applying EAST text detector [19] followed by OCR [20] on the extracted text regions. However, DL architecture diagrams contain specific words or acronyms which may not be available in general dictionary. Thus, of-the-shelf OCR performs poorly. To improve performance, we created a DL dictionary with possible words/acronyms found in DL diagrams from the training dataset.

Arrow Detection: Node connections in terms of arrows in DL diagram give important information about the flow of the entire DL model design. To detect the arrows, all the detected nodes and the text regions are masked out from a figure. Next, Hough line detection algorithm is applied to detect the arrow lines. The direction of the arrow is obtained by analyzing the pixel distribution of the contour corresponding to a detected arrow line. Figure 2 shows one example image after extraction of all the nodes, texts and arrows.

2.4. Structural Analysis for Graph Generation

Structural analysis is performed to find spatial and logical relations among the nodes, text, and arrow candidates to generate the final representation in terms of DKG. General

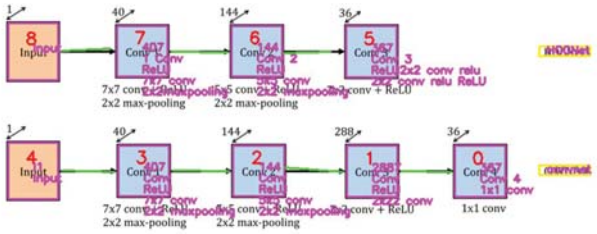

Fig. 2: All the nodes extracted from an example DL diagram are shown here. The node numbers are colored in red. The text description associated with a node is depicted with pink color within the node. The text box in yellow color shows text description not tagged with any specific node. The connectivity among nodes are shown in green color.

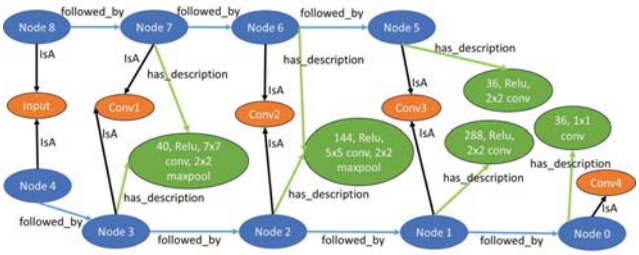

Fig. 3: Graph extracted from Figure 2.

flowchart representing a workflow or process or algorithm follows strict structural grammar [8]. Existing flow diagram analysis methods [8, 10] utilize this structure grammar to perform structural analysis on traditional flow charts. As DL diagrams do not strictly follow such structures, e.g. each two node may not necessarily be connected with an arrow, it is not possible to apply existing flow diagram analysis methodologies [21, 9, 22, 23, 24] to decode a DL diagram. Instead, we employ DL specific structure grammar as described next to perform flow detection.

The node, text and arrow contours are first sorted based on the location and direction. The nodes generally contains some DL layer description. According to [18], a layer description is generally provided within a detected block or in its vicinity. So, each text box is tagged with the nearest neighbor node according to a pre-defined threshold. Then, based on the text description, the type of the node is identified. However, if the nearest neighbor node is beyond the pre-defined threshold, the text is not tagged with any node and kept as an independent text component. In such cases, it has been observed that the text generally serves as image title or image section title [25].

Next, the start and end points of the arrows are analyzed to detect connectivity between two nodes/layers. We create a grammar defining a list of possible next layers for a given current layer. The arrow directions along with possible valid next node/layer information is used to identify valid flow. This grammar also helps to find the flow among a set of nodes put in vertical or horizontal order without using any arrow. After detecting DL design flow, a deep knowledge graph is created with following relationships describing a diagram: (i)"$found\_in$" (⟨FigureID-$found\_in$-paper title⟩), (ii) "$has\_caption$" (⟨FigureID-$has\_caption$-figure caption⟩), (iii) "isA" (⟨ComponentID/TextID-isA-layer name⟩), (iv) "$has\_description$" (⟨ComponentID/TextID-$has\_description$-layer properties⟩), (v) "$followed\_by$": ⟨ComponentID/TextID-$followed\_by$-ComponentID/TextID⟩, (vi) "has_flow": ⟨FigureID-has_flow-flow direction⟩).

Figure 3 shows the diagram graph constructed from Figure 2. This representation is the foundation of the RDF [2] DKG creation, known as the standard for representing connected semantic content. These knowledge graphs are stored in triple stores and the information is queried from them using triple pattern queries.

3. RESULTS

3.1. Dataset

We downloaded 1750 papers from arXiv.org papers covering five different research areas (CVPR, ICML, ECCV, ICCV, NIPS) using "deep learning" as the input query. 14,351 figures were extracted from these papers using [13] with 94% precision at 90% recall. Out of the these figures, 4488 figures were identified to portray DL design flow and the remaining 9863 figures were other type of figures. Of all the figures annotated as 2D-box, we randomly sampled over 150 figures for further detailed annotations, i.e., nodes, arrows, text, relations, etc. Annotating the figures yielded 1272 nodes, 2183 arrows, 5555 test boxes and 5555 relations. A substantial fraction of the object annotations has overlapping bounding boxes and complex connectivity/relationships. The dataset contains an average of 67 relationships per image.

Figure Type Classification Accuracy: Labeled image set is split into training, validation, and test set with 8:1:1 ratio to train the deep neural network classifiers. Accuracy of the classifiers is reported in Table 1. It can be observed that the DL diagrams can be identified with 95% accuracy on the test dataset which is better than the 86% accuracy [18]. Further, on highly varying DL flow design images, multi-class classifier obtained F1 score more than 70%.

Table 1: Performance of figure type classifiers.

Type	Approach	Precision	Recall	F1-score	Accuracy
Binary Classifier	VGG19	0.86	0.86	0.86	0.94
	Resnet	0.86	0.86	0.86	0.95
Multi-cls Classifier	VGG19	0.85	0.62	0.70	0.62
	Resnet	0.87	0.66	0.71	0.66

Content Extraction Accuracy: Evaluating figure analysis results is a challenging endeavor as it demands detailed annotation of the figures within research papers. Therefore, most previous works have restricted their evaluation to manual inspection [26, 27]. The availability of our detailed annotated dataset allows thorough analyses of the various components of our approach. *Node detection* accuracy is measured by using the standard bounding box overlap-criteria from object detection [28]. More specifically, we regard a predicted bounding box BB_p for the node box to be correct if its

intersection-over-union (IoU) with the ground-truth box BB_g is above 0.5, i.e., $\frac{BB_p \cap BB_g}{BB_p \cup BB_g} > 0.5$. Under this metric, we obtained an accuracy of 69.44%. Qualitative analysis of the results reveal that the node detection module was not performing well if the components are having irregular/fuzzy/dotted border, color-filled box with color transition within the box. *Text detection* accuracy is measured independently based on the text box position detection, text identification modules. Text box position detection accuracy was found to be 62.84% using the IoU criteria. It was observed that the text detection module may not work well in challenging cases of low-resolution text, equations or symbols. Text identification accuracy without DL dictionary was found to be 51.6%. The accuracy improved to 72.4% after use of DL dictionary.

Graph Construction: While the above evaluations reveal the component-level performance, we also evaluated our overall figure analysis performance in terms of relationship detection. The accuracy metric measures the fraction of ground-truth relationship triplets (subject-predicate-object) that appear among the top 1 most confident triplet predictions in an image. The choice of this metric is, as explained in [29], due to the sparsity of the relationship annotations in our dataset-metrics like mAP would falsely penalize positive predictions on unlabeled relationships. Accuracy of our model for diagram-to-graph generation task was found to be 59.86%. Note that several components need to be sequentially accurate for the entire parsing to be considered correct. A substantial fraction of missed detection originates from lower accuracy of connectivity detection module and text detection module. For example, if the arrow heads are not detected properly, the "$followed\_by$" relationship will not be detected accurately. Similarly, if the text parsing module does not generate correct output, "isA" predicate shows wrong results. Our approach does an impressive job despite the high structural variations in the figure as well as the presence of heavy clutter in the diagram figure.

3.2. Applications: Novelty Detection

The DKG created by the proposed Diag2Graph pipeline enables a variety of exciting applications, like DL architecture question answering, novelty detection, information retrieval and recommendation systems. Here we demonstrate how the DKG can be used for novelty detection by extracting the information into an intermediate form that can capture the high-level semantics from DL publications. To do this, we represent the DKG using low-dimensional latent vectors that captures the semantic properties of a DKG by representing it with its features distributed across multiple vector components.

We leverage Graph AutoEncoder (GAE) [30] to embed the KG nodes to a lower dimensional space in such a way that certain proximity measures are preserved. We introduce a super-node for each subgraph DKG that is connected to all the nodes in that subgraph and serve as subgraph representative.

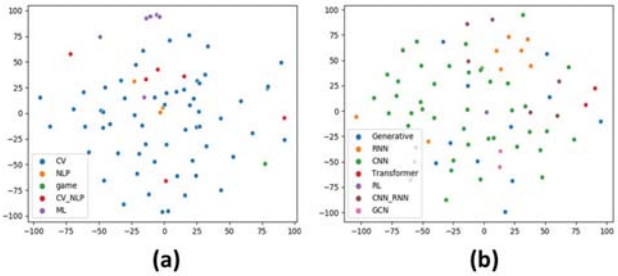

Fig. 4: DKG embeddings from images of 143 papers. The colors indicate: (a) area of research, (b) type of DL network.

Thus, when embedding is done, the super-node embedding can be seen as a summary of the subgraph. Once the learning process is complete, the embedding space can be used as feature inputs to perform various machine learning tasks, like predicting the group that a node belongs to.

We construct graphs for 213 DL architecture images from 143 papers comprising of 4,252 nodes with 393 features per node and 16,098 edges/relations between them. For the visualization purpose, we take the embedding of the super-nodes and apply TSNE [31] to generate the images shown in Figure 4. It can be observed from Figure 4(a) that all Recurrent Neural Network (RNN) diagrams (orange color in top right corner) are clustered together showing their architectural similarity. Proximity of CNN_RNN type diagrams (brown color) to the RNN cluster reveals their resemblance to RNN type architecture. On the other hand, Figure 4(b) reflects diagram similarity in terms of research area. High similarity score between two papers in terms of research area as well as diagrams type calculated from the features embedding indicates probability of limited novelty that needs further close investigation.

4. CONCLUSIONS

In this paper, we introduced Diag2Graph, an end-to-end framework for parsing deep learning diagram figure in a research paper and representing it as a knowledge graph. This is the first work to the best of our knowledge that represents diagrams as knowledge graphs to enable rich indexing and search of DL architecture content in future. Our experimental analysis has confirmed that figure parsing in scholarly big data is a challenging vision application. The Diag2Graph generator currently suffers from successfully parsing the diagram in presence of heavy clutter showing complex overlapping node connections. Techniques from vascular tracking [32] could be applicable here. Legend parsing to interpret color codes in arrows and nodes is currently not supported. Finally, our DL diagram analysis approach models and trains the different components (nodes, arrows, and text-data) independently. Jointly modeling all the components and training them together within an end-to-end deep network is an exciting endeavor. While our current framework is generalizable for parsing a variety of flow diagram-figures, it has only scratched the surface with interesting open challenges ahead.

5. REFERENCES

[1] Jean Charbonnier, Lucia Sohmen, John Rothman, Birte Rohden, and Christian Wartena, "Noa: a search engine for reusable scientific images beyond the life sciences," in *European Conference on Information Retrieval*. Springer, 2018, pp. 797–800.

[2] World Wide Web Consortium et al., "Rdf 1.1 concepts and abstract syntax," 2014.

[3] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 697–706.

[4] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer, "Dbpedia spotlight: shedding light on the web of documents," in *Proceedings of the 7th international conference on semantic systems*, 2011, pp. 1–8.

[5] Jian Wu, Kyle Mark Williams, Hung-Hsuan Chen, Madian Khabsa, Cornelia Caragea, Suppawong Tuarob, Alexander G Ororbia, Douglas Jordan, Prasenjit Mitra, and C Lee Giles, "Citeseerx: Ai in a digital library search engine," *AI Magazine*, vol. 36, no. 3, pp. 35–48, 2015.

[6] Po-shen Lee, Jevin D West, and Bill Howe, "Viziometrix: A platform for analyzing the visual information in big scholarly data," in *Proceedings of the 25th international conference companion on world wide web*, 2016, pp. 413–418.

[7] Noah Siegel, Zachary Horvitz, Roie Levin, Santosh Divvala, and Ali Farhadi, "Figureseer: Parsing result-figures in research papers," in *European Conference on Computer Vision*. Springer, 2016, pp. 664–680.

[8] Dorothea Blostein, "General diagram-recognition methodologies," in *International Workshop on Graphics Recognition*. Springer, 1995, pp. 106–122.

[9] Martin Bresler, Daniel Prusa, and Vaclav Hlavac, "Online recognition of sketched arrow-connected diagrams," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 19, no. 3, pp. 253–267, 2016.

[10] Chengcheng Wang, Harold Mouchère, Aurélie Lemaitre, and Christian Viard-Gaudin, "Online flowchart understanding by combining max-margin markov random field with grammatical analysis," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 20, no. 2, pp. 123–136, 2017.

[11] Tobias Kuhn, ThaiBinh Luong, and Michael Krauthammer, "Finding and accessing diagrams in biomedical publications," in *AMIA Annual Symposium Proceedings*. American Medical Informatics Association, 2012, vol. 2012, p. 468.

[12] Sagnik Ray Choudhury, Prasenjit Mitra, and Clyde Lee Giles, "Automatic extraction of figures from scholarly documents," in *Proceedings of the 2015 ACM Symposium on Document Engineering*, 2015, pp. 47–50.

[13] Christopher Clark and Santosh Divvala, "Pdffigures 2.0: Mining figures from research papers," in *2016 IEEE/ACM Joint Conference on Digital Libraries (JCDL)*. IEEE, 2016, pp. 143–152.

[14] Sagnik Ray Choudhury and Clyde Lee Giles, "An architecture for information extraction from figures in digital libraries," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 667–672.

[15] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[17] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al., "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[18] Akshay Sethi, Anush Sankaran, Naveen Panwar, Shreya Khare, and Senthil Mani, "Dlpaper2code: Auto-generation of code from deep learning research papers," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[19] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang, "East: an efficient and accurate scene text detector," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 5551–5560.

[20] https://github.com/tesseract ocr/tesseract, "Tesseract open source ocr engine," .

[21] Kurmanbek Kaiyrbekov and Metin Sezgin, "Stroke-based sketched symbol reconstruction and segmentation," *IEEE Computer Graphics and Applications*, 2019.

[22] Rumaan Bashir and Kaiser J Giri, "Diagram recognition: Domain knowledge based approach," in *2013 International Conference on Machine Intelligence and Research Advancement*. IEEE, 2013, pp. 445–449.

[23] Caglar Tirkaz, Berrin Yanikoglu, and T Metin Sezgin, "Sketched symbol recognition with auto-completion," *Pattern Recognition*, vol. 45, no. 11, pp. 3926–3937, 2012.

[24] Xavier Muñoz Pujol et al., *Image segmentation integrating colour, texture and boundary information*, Universitat de Girona, 2003.

[25] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi, "A diagram is worth a dozen images," in *European Conference on Computer Vision*. Springer, 2016, pp. 235–251.

[26] Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer, "Revision: Automated classification, analysis and redesign of chart images," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 393–402.

[27] Peng Wu, Sandra Carberry, Stephanie Elzer, and Daniel Chester, "Recognizing the intended message of line graphs," in *International Conference on Theory and Application of Diagrams*. Springer, 2010, pp. 220–234.

[28] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[29] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei, "Visual relationship detection with language priors," in *European conference on computer vision*. Springer, 2016, pp. 852–869.

[30] Thomas N Kipf and Max Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[31] Laurens van der Maaten and Geoffrey Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[32] Amos Sironi, Vincent Lepetit, and Pascal Fua, "Multiscale centerline detection by learning a scale-space distance transform," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2697–2704.