DEEP-LEARNING FEATURES, GRAPHS AND SCENE UNDERSTANDING

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Engineering by Research

by

ABHIJEET KUMAR
201302197
abhijeet.kumar@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
JANUARY 2020

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled " **DEEP-LEARNING FEATURES, GRAPHS AND SCENE UNDERSTANDING**" by Abhijeet Kumar, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Dr. AVINASH SHARMA
Center for Visual Information Technology,
Kohli Center on Intelligent Systems,
IIIT Hyderabad

To Time

Acknowledgments

As I submit my thesis, I extend my gratitude to all those people who helped me in successfully completing this journey.

Foremost my deepest gratitude to Prof. Avinash Sharma and Prof. Madhav Krishna for their constant support and encouragement to explore and chart my own research path. Im really fond of my stints as a TA under Prof. Naresh Manwani and Prof. Ravi Kiran. I would also like to extend my sincerest thanks to Prof. Vineet Gandhi and Prof. Vineet Balasubramian with whom my collaborations did not bore fruit.

I would also show my gratitude to the CVIT ecosystems, through which I came to have multiple technical/non-technical discussions with wide background of students in PhD/MS/Honors and also developed an understanding of the vast variety of problems begin attempted at the centre. The names in this list are too many and it is almost impossible to enumerate all of them.

I will also like to thank my parents and my brother whose support has been always immense in all my adventures in life. I would like to thanks my friends (especially Rohit, Noor, Anurag, Yash, Anchal, Abbhinav, Govinda and Anuj) at IIIT-Hyderabad who have provided motivation, philosophical discussions, *Dullah* Talks etc. I also appreciate the tolerant nature of my co-authors (Gunshi Gupta and Anjali Shenoy).

I will always look back to enjoy the fond memories I have of this institute.

Abstract

Scene Understanding has been a major aspiration of computer vision from its early days. Its root lies in enabling the computer/robot/machine to understand, interpret and manipulate visual data, in similarity to what an average human eye does in front of a natural/artificial localized location/scene. This ennoblement of the machine have a widespread impact ranging from Surveillance, Aerial Imaging, Autonomous Navigation, Smart Cities and thus scene understanding have remained as an active area of research in the last decade.

In the last decade, the scope of problems in the scene understanding community has broadened from Image Annotation, Image Captioning, Image Segmentation to Object Detection, Dense Image Captioning, Instance Segmentation etc. Advanced problems like Autonomous Navigation, Panoptic Segmentation, Video Summarization, Multi-Person Tracking in Crowded Scenes have also surfaced in this arena and are being vigorously attempted. Deep Learning has played a major role in this advancement/development. The performance metrics in some of these tasks have more than tripled in the last decade itself but these tasks remain far from solved. Success originating from deep learning can be attributed to the learned features. In simple words, features learned from a Convolutional Neural Network trained for annotation are in general far more suited for captioning then a non-deep learning method trained for captioning.

Taking cue from this particular deep learning trend, we dived into the domain of scene understanding with the focus on utilization of prelearned-features from other similar domains. We focus on two tasks in particular: Automatic (multi-label)Image Annotation and (Road)Intersection Recognition. Automatic image annotation is one of the earliest problems in scene understanding and refers to the task of assigning (multiple) labels to an image based on its content. Whereas intersection recognition is the outcome of the new era of problems in scene understanding and it refers to the task of identifying an intersection from varied viewpoints in varied weather and lighting conditions. We focused on this significantly varied task approach to broaden the scope and generalizing capability of the results we compute. Both image annotation and intersection recognition pose some common challenges such as occlusion, perspective variations, distortions etc.

While focusing on the image annotation task we further narrowed our domain by focusing on graph based methods. We again chose two different paradigms: a multiple kernel learning based non-deep learning approach and a deep-learning based approach, with a focus on bringing out contrast again. Through quantitative and qualitative results we show slightly boosted performance from the above mentioned paradigms. The intersection recognition task is relatively new in the field. Most of the work in field focuses on Places Recognition which utilized only single images. We focus on temporal information i.e. the traversal of the intersection/places as seen from a camera mounted on a vehicle.

Through experiments we show a performance boost in intersection recognition from the inclusion of temporal information.

Contents

List of Figures

List of Tables

Chapter *1*

Introduction

Scene understanding has been an important goal of computer vision systems - to enable machines identify, interpret and manipulate objects, surfaces and their interactions in images and videos and interpret relationships between them. Tackling/Attempting problems from the domains of scene understanding entails compatible data. For example learning to identify faces requires a corpus of face dataset. The widespread outburst of digital data, enabled by ever speeding network services, increasing smart electronic devices(mobiles, tablets) and fast-paced lifestyle, has led to a steep rise in data-driven technologies and has opened up a scope for vast number of complex problems. A decade ago problems in scene understanding consisted of image-captioning, image annotation, object detection etc while the current focus is on problems 3D reconstruction from 2D images, panoptic segmentation, multi-person tracking in crowds, very long video summarization. Currently, scene understanding is in the core of many imminent successes of modern technology - autonomous driving, human-machine interaction, remote sensing, among several others. Deep Learning, a subset of machine learning, has been shown to benefit from huge amounts of data and has been an active area of research. Currently models based on deep-learning forms state of the art for vast array of problems.

Our focus in this thesis derives both from the paradigm shift in the complexity of the problems in the domain of scene understanding and the success of deep learning(specifically deep learning features). To limit the scope of our work, we select two problems from scene understanding (i) Multi-label image annotation and (ii) Intersection recognition in road junctions. Multi-label image annotation can be categorized into a traditional scene understanding problem, whereas intersection recognition can be categorized into modern scene understanding problems. Our work in the image annotation domain focuses on graph based learning. Here we have focused on two specific directions, Chapter 3 focuses on traditional(non-deep learning) graph based approach whereas Chapter 4 focuses on graph deep learning based approach. We also utilize deep learning features in our work. Specifically, we use pre-trained deep learning models to extract representations for images which we use later in our models. Apart from deep-learning features Chapters 4 and 5 also utilizes state of the art deep learning architectures.

Now we brief the organization of this chapter. We first detail the motivations and problem definitions for the task of multi-label image annotation and intersection recognition. We then discuss the associated

challenges in these tasks and lay down our contribution in this thesis. We finally provide an overview of the remainder of the chapters in this thesis.

1.1 Multi-Label Image Annotation

The goal of the multi-label image annotation task is to assign all semantically meaningful labels[1] to an image. Multi-label image annotation has applications in nearly every application domain of computer vision ranging from object detection, instance segmentation, captioning, robotics, remote sensing, autonomous driving, surveillance, medical imaging etc. Multi-label image annotation is one of the keystone of scene understanding.

The term "semantically meaningful" could lead to a very large number of labels for any image. Hence to limit this set of labels, a label-set or dictionary of labels/tags/words to chose from is also provided. Tagging all the relevant words from this label-set/dictionary is the practical scenario of the task at hand. Its easily observable the the visual content of an image captures a wide variety of semantics at multiple levels of granularity. Figure 1.1 shows an example from MSCOCO dataset on the left and a dictionary of words on the right(also indicated is their presence/absence in the image).

Multi-label image annotation is a variant of single-label image annotation (commonly known as Image Annotation) where each image is tagged with just one relevant label from a dictionary set. It is imperative to follow that single-label image annotation closely resembles to multi-label image annotation but simple extensions to single-label task do not perform as good on the multi-label task. This argument is validated from observing the current state of the arts in two tasks:- while single-label image annotation is close to being solved (a top-5 accuracy of 98% on the Image-Net dataset is already achieved), multi-label image annotation is far from solved(F1-score of around 75% on the MSCOCO dataset). We propose two graph based models (one with a deep-learning setup and another without a deep learning setup) for this task. We elaborate on these models, their specific backgrounds, datasets used and results in Chapter 3 and Chapter 4.

1.2 Intersection/Junction Recognition

A road intersection or a road junction [2] typically is a meeting point of three or four road segments. In this problem, we focus on recognizing weather two given videos consist of same geographical location while only using a monocular camera mounted on the top of a vehicle. A road intersection is hard to recognize as the approach to it could be from any of the 3 or 4 road segments its composed of, each of which have a different view(backgrounds are completely different) of the intersection. Figure 1.2 depicts the scenario of an intersection and two traversals(in red and green) of this intersection.

[1] We refer the words *labels* and *tag* interchangeably

[2] We refer the words *junction* and *intersection* interchangeably

(a) An image from MSCOCO dataset

(b) An example of a dictionary of words. The tick/wrong marks indicate weather the label/tag is present in the image on the left

Figure 1.1: Multi-label image annotation problem setup

Recognizing an intersection from a different approach sequence or a sequence of viewpoints different from those seen before can be pivotal in various applications that include autonomous driving, driver assistance systems, loop detection for SLAM including multi-robot and multi-session SLAM frameworks. Intersection recognition from disparate video streams or image sequences is an extremely challenging problem that stems from the large variations in viewpoint, weather and appearance across traversals. Complexity also emanates from varying levels of traffic and chaos at a junction, as well as due to changing levels of occlusion, illumination between two video sequences of a specific junction. Lack of annotated datasets on intersections also poses a challenge.

Intersections being critical points in road based navigation, are likely to be points of crossover between trajectories of multiple agents/cars. Owing to their four-way structure, imagery captured in different traversals of junctions depicts very different views and alignment of common landmarks unlike what is observed in place recognition datasets(e.g., [68]). If agents can recognize a previously visited place better in such tricky scenarios, they are endowed with better localization capability. Intersection recognition also closely follows intersection detection [10], wherein the immediate question to answer upon detecting an intersection is if the detected intersection is the same as one seen previously. We proposed a novel LSTM based Siamese style deep network model for this problem while showing competitive results on real and synthetic datasets. We elaborate on this model, datasets for this task and results in Chapter 5.

1.3 Associated Challenges

A wide variety of challenges can be associated with scene recognition. We provide a small brief of some of these challenges in this section.

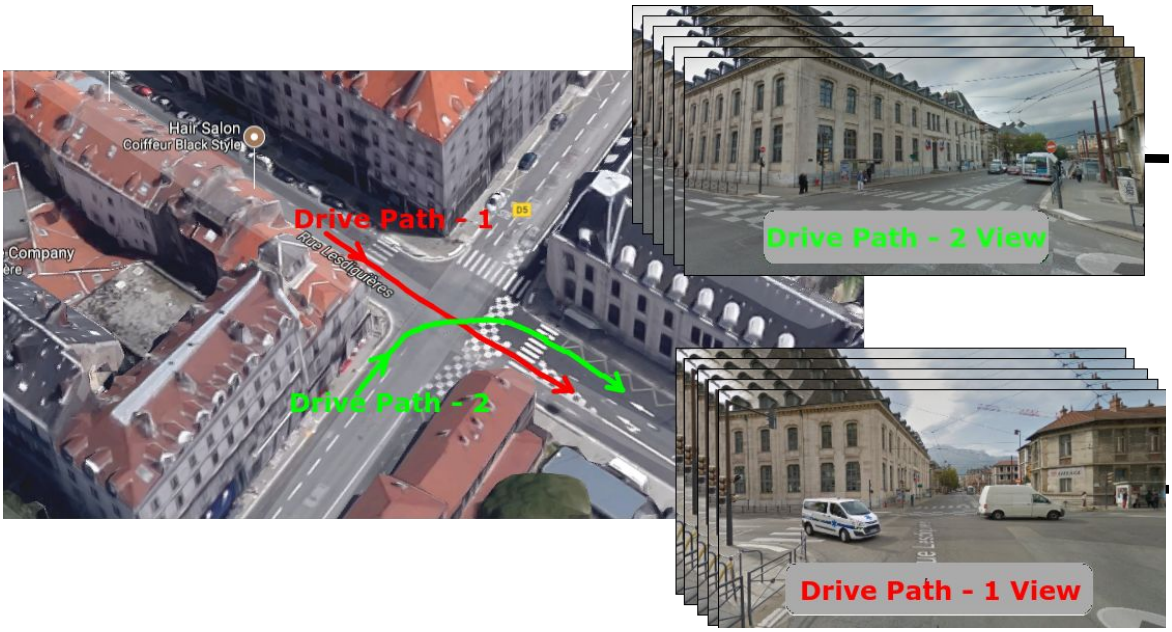


Figure 1.2: Intersection recognition problem setup: Given two videos consisting of road *traversal*(in red and green) of intersection, identify if they belong to the same geographical place/location.

1. **Illumination Changes**: Lighting conditions in outdoor images vary drastically. They may range from very high to very low foreground-background contrast. This might result in highlighting random objects in the scene. Indoor surroundings and illumination can be manipulated to make the object of interest in focus with respect to the background. Illumination changes usually results in challenging problems for many computer vision applications such as recognition, tracking and motion analysis. Figure 1.3a shows varying illumination conditions highlighting the background or the foreground obscuring visibility.

2. **Occlusion**: In general, outdoor images have more than one object or the object interacts with elements of the surroundings. This leads to occlusion by some other object like a pole, building etc. These cases like partial or full occlusion create uncertainty in determining the location of object which are not visible. Apart from the above mentioned occlusions which were caused by an outside entity (whether living or non-living object), the object of interest can occlude itself, which is termed as self occlusion. Figure 1.3b shows such examples of occlusion.

3. **Dataset Bias**: Each image can be tagged with multiple labels from the given dictionary but in general some labels appear more frequently then others. In-fact the occurrence pattern of labels follows the long-tail curve. This implies that a few labels appear most of time and the rest occurs only a few times in the presented dataset. This includes a heavy bias in the dataset which needs to controlled in order to make recognition unbiased.

(a)

(b) (c)

Figure 1.3: Challenges: (a) Illumination Changes, (b) Occlusion, (c) Dynamic Background. Images are taken from Mapillary and Google Images.

4. **Incomplete Labelling**: Creation of a dataset involves manual labelling of images. This tasks is done by human annotators who are not good at these mundane tasks. This may result in images not being tagged with all the potential labels. Figure 1.1 shows such an example where the image can also be labelled with *grass*. To curb these error, multiple annotators label the images in more recent datasets.

5. **Dynamic Background**: Some parts of the scene may contain movement (a fountain, movement of other objects, the swaying of tree branches, water waves etc.), but should be regarded as background, according to their relevance. Such movement can be periodical or irregular (e.g., traffic lights, waving trees). Handling such background dynamics is a challenging task. Figure 1.3c shows examples of dynamic background having water waves from the fountain.

6. **Weather Change**: Outdoor scenes may look very different during different seasons/weathers. While a foggy/rainy day may hinder capturing the background while a sunny day will have better contrast and shinier objects. Figure 1.4 (a) shows an image of on a rainy and a summer day.

7. **Object Speed**: The speed of the moving object plays an important role in its detection. Intermittent motions of objects cause ghosting artifacts in the detected motion, i.e., objects move, then stop for a short while, after which they start moving again. There may be situations in a video, where the still objects suddenly start moving, e.g., a parked vehicle driving away, and also abandoned objects.

8. **Shape and Structural changes**: A single object can have many different structural possibilities which makes a direct modelling of an object in general near impossible. Similarly, non-rigid objects can change shape when forces are applied on them. An example of this could be the pose of a human when he/she is moving. Figure 1.4b shows an example images of chair which are structurally different.

9. **View Variations**: Different camera positions may capture the same geographical location but the corresponding images are likely to different. One such possibility arises when an image captures the front of a building while the other captures the back-view of the same building. In such case backgrounds of the two images may overlap partially or not overlap at all. Figure 1.4a shows an example where the *Eiffel Tower* is captured from two different viewpoints.

(a)

(b)

Figure 1.4: Challenges: (a) Weather Changes, (b) Structural Changes

1.4 Our Contributions

The main contribution of this thesis are:

1. **Multi-label image annotation** We introduced the notion of *neighborhood-types* and proposed a new label histogram characterization of image-neighborhoods. The notion of *neighborhood-types* is based on hypothesis that similar images in content/feature space should also have overlapping neighborhoods. Building upon this we used diffusion as label transfer mechanism to associate tags and derived a closed from solution.
   We also explored Graph Convolution Neural Networks for multi-label image annotation where we propose a graph based deep-learning solution for the problem of multi-label classification. Through extensive experimentation of our architecture, we explore the effects of graph formation and the input signals on the performance.

2. **Intersection/Junction recognition** We proposed a novel stacked deep network ensemble architecture that combines state-of-the-art CNN, bidirectional LSTM and Siamese style distance function learning for the task of view-invariant intersection recognition in videos. We have collected annotated data of around 2000 videos from GTA gaming platform and Mapillary consisting of varied weather conditions and varied viewpoints.

1.5 Thesis Overview

Now we provide the organizational scheme of the upcoming chapters in this thesis. The outline of the following chapters is:

- Chapter 2 provides the literature survey for image-annotation, intersection recognition and deep learning in graphs.
- Chapter 3 details out our approach for image annotation via diffusion combined with Multiple Kernel Learning (MKL) model.
- Chapter 4 describes our approach, experiments and results using graph based deep learning model for image annotation.
- Chapter 5 dives into the problem of intersection recognition.
- Finally chapter 6 summarizes the main contributions of the thesis.

Furthermore, each of Chapter 3 and Chapter 4 is broken down into (i) Motivation, Brief and Contributions (ii) Proposed Approach (iii) Dataset (iv) Experiments and (v) Results and Discussions.

Chapter *2*

Related Works

2.1 Image Annotation

Over the last two decades, a plethora of methods have appeared in this area. We brief about a lot of these methods while our focus is on most relevant, popular and recent methods. We try to categorize the previous works into 5 categories based on the underlying model's ideology.

2.1.1 Generative, Discriminative and Hybrid Models

Sahin et al. [25] model the problem from a machine translation perspective where a region is mapped to a label. Relevance Models [27, 53, 42] model the problem as joint probability distribution between image features and labels. Multiple Bernoulli Relevance Model(MBRM) [27] proposed regular region blocks for features and a multiple Bernoulli model using non-parametric kernel density estimate. Topic Models [8, 70, 25, 101] uses probabilistic method to represent the joint distribution of visual and textual features by maximizing the generative data likelihood.

Xinag et al. [101] proposed a Markov Random Field model, which captured many previously proposed generative models, but had an expensive training step as it learnt an MRF per label. Discriminative models were proposed in [105, 33, 95, 28]. These methods learn label-specific models to classify an image belonging to the particular label. However they fail to capture label-to-label correlations. A hybrid model was presented in [71] combining a generative [27] and discriminative (SVM) model aimed at improving the number of labels recalled.

2.1.2 Nearest Neighbor Approaches

Though simple and highly intuitive, NN methods are among the best performing ones. [34] introduced metric learning to fuse an array of low-level features. They used cross entropy loss in addition to weighted (based on distance or rank) label propagation. Recently, [96] defined a Bayesian approach with two pass kNN for addressing the class-imbalance challenge and subsequently used an extension of existing LMNN approach [100] as metric learning to fuse different feature sets. One major limitation of these

approaches is that they are global methods and heavily rely on metric learning construct, which makes it difficult to scale them to large datasets.

Alternatively, a local variant of NN method proposed in [45] performs the Non-negative Matrix Factorization (NMF [54]) of features from images in a smaller neighborhood, which are made to follow a consensus constraint. Here, the class-imbalance is dealt by means of weighting different feature matrices. However, this is purely a local approach and hence fails to capture the global structure of the latent space.

Recently, [92] and [96] report performance improvement over the NN methods by using cross modal embedding such as Canonical Correlation Analysis (CCA) or Kernel Canonical Correlation Analysis (KCCA). This is again an attempt to learn global common latent space. However selection of an appropriate kernel function and scalability of KCCA poses a major challenge in these methods.

2.1.3 Graph Based Models

A graph based transductive method for explicitly capturing label-to-label and image-to-image similarities was proposed in [97]. Recently, [73] proposed a hypergraph diffusion based transductive method and exploited multi-scale diffusion to address the class-imbalance problem. However, both these methods are semi-supervised in the sense that they need access to all test data for prediction. Additionally the hypergraph diffusion method is not scalable due to requirement of storage and computation of eigen-decomposition of very large matrices.

2.1.4 Deep Learning Based Methods

Inspired by the recent success of deep neural net architectures in image classification [88, 86], different approaches involving deep nets have been tried for multi-label classification [40, 67, 99, 64, 102, 35, 107, 15]. [40] modeled these relationship on a hierarchical model exploiting Long Short Term Memory (LSTM) by incorporating inter-level and intra-level label correlations which were parsed using WordNet. CNN-RNN [98] learns a joint image-to-label embedding and label co-occurrence model in an end-to-end way. Semantically Regularised CNN-RNN (S-CNN-RNN) [62] improves on the CNN-RNN model by using a semantically regularized embedding layer as an interface between the CNN and RNN which enables RNN to capture the relational model alone. [43] showed that the order of label prediction mattered while using RNN (or LSTM) for label modelling.

[44] proposed exploiting image metadata to generate neighbors of an image and blend visual information using neural-nets. Recent works in Deep nets capture label-to-label relationships more explicitly than before. Another recent work in [72] converted labels to a word2vec vector [67] and performed label transfer using nearest neighbor methods in embedding space computed using CCA or KCCA. More recently visual attention models [35, 64, 99] have been on the rise too.

2.1.5 Graph Based Deep Learning Models

Recently deep-learning methods have been introduced in the context of graphs [24, 38, 80, 58]. Gated Graph Neural Network (GGNN) [58] is a LSTM variant for graphs which learns a propagation model that transfers information between nodes depending on the edge types. [66] introduced Graph Search Neural Network (GSNN) which improves GGNN [58] by diminishing the computational issues. GSNN is able to reason about the concepts by capturing the information flow between nodes in the noisy knowledge-graphs. GSNN is different from our model in the propagation modeling in graphs. In Chapter 4, we explicit model the label propagation with diffusion framework on graphs while GSNN learns the network propagation parameters in the knowledge-graph.

Contemporary to this submission a recent graph based method [18] have been published. Similar to our work in Chapter 4, Chen et al. [18] models label patterns/co-relations utilizing graph convolution networks, but they use the formulation from [48]. They also differ in the way they model Joint Space (refer Subsection 4.2.3) and their formulation of the knowledge graph matrix (refer Subsection 4.2.1.1). They utilize work-embedding as signals on nodes of the graph while we use Faster-RCNN [77] output detections as signals.

2.2 Intersection Recognition

In existing literature around robotic perception, there are limited efforts towards intersection detection and recognition. Some of these use sensors other than cameras such as laser and virtual cylindrical scanners [108, 55].

Within the robotics community, loop detection methods have used diverse image recognition and retrieval techniques [32, 29, 5] that have not attempted to detect loops from very diverse viewpoints, though they perform admirably in the presence of weather changes [61] or under the duress of varying traffic [36]. In the vision community, CNN [87] features have been used to efficiently compare scenes or structures which are viewed from varying distances, which give a zoom-in vis-a-vis zoom-out effect with minor changes in viewing angles.

Recently, a framework for detecting intersections from videos was proposed in [10]. They used LSTM based architecture to observe the relative change in outdoor features to detect intersection. Nevertheless, their work did not focus on the recognition task. On the other hand, visual loop closure detection techniques in the literature [6, 32, 29] mainly focused on image level comparison of scenes. These techniques try to find re-occurring scene during the driving session based on fast KD tree based comparison of image features. Although these methods achieve admirable accuracy for loop detection they cannot be improvised for view-invariant recognition over videos.

In recent years, gaming environments have been created and/or used for generating and annotating datasets. [78] used Grand Theft Auto V (GTA) gaming-engine to create large-scale pixel-accurate ground truth data for training semantic segmentation systems. SYNTHIA [79] is a virtual world environment which was created to automatically generate realistic synthetic images with pixel-level annotations for the

task of semantic segmentation. [78] and [79] show the added improvement in the real-world setting by using synthetic-datasets for training deep-network models. CARLA [23] is another open-source simulator developed to facilitate learning and testing of Autonomous Driving algorithms. GTA environment is more visually realistic than SYNTHIA and CARLA environments.

Real-world datasets such as [69, 1] are designed with the idea of visual place recognition using images. These datasets are limited in the pathways covered and the number of intersections. Another real world street-level imagery platform [4] consist of images and image-sequences with variability in the pathways and intersections and thus enables sequence learning for intersection recognition. Concurrent works have used this platform to create Mapillary Vistas Dataset [74] which is a semantic segmentation dataset consisting of 25k images. We have used Mapillary dataset to showcase our result on real world scenarios.

Recently [81] and [31] have attempted visual localization under drastic viewpoint changes. [81] needs various modes of data including: semantic segmentations, RGBD images and camera parameters. Its a computationally expensive pipeline focused on relocalization, involving 3D voxel grid constructions of the entire previous traversal sequence and of the query sequence, followed by a matching and pose estimation procedure. Due to key differences in problem context and usage of multimodal inputs, we omit a quantitaive comparison with [81]. [31] proposes graph construction from semantically segmented input data with graph matching using random walk descriptors. We implement and compare against this method in a limited setup in our baseline experiments in Section 5.2.1. Specifically there has been no prior effort towards recognizing intersections when approached from different road segments that constitute the intersection.

2.3 Graphs Deep Learning

In this section we first introduce a graph with notations. Then we provide the motivation for deep learning in graphs. Then we mention a few existing works on networks on graph while paralelly following a taxonomical categorization of graph networks. We then discuss in detail the spectral graph theory and graph signal processing which forms a basis for understanding convolution operator in graphs. We then move exclusively to graph convolution networks and provide an overview.

A graph is defined as a set of elements and their pairwise connections. Mathematically, the set of elements are defined as nodes or vertices and connections are known as edges. The set of nodes is denoted as $V = (v_1, \cdots, v_n)$ and set of edges as $E = \{(v_i, v_j)|(v_j \sim v_i) = w_{i,j} \geq 0\}$. An edge exists only if two nodes have a relationship, $w_{i,j}$ is defined as the weight of the relationship between nodes v_i, and v_j. Graphs are categorized as non-euclidean data as the pairwise relationship may not necessarily be an euclidean space distance of nodes. An edge can represent many sorts of relationships; similarity, dis-similarity, or a constraint. And weights can be used to define the magnitude of this relationship. A graph can be represented by a matrix, called weighted adjacency matrix, $\boldsymbol{W}_{n\times n}$ whose $(i,j)^{th}$ entry $w_{i,j}$ is the similarity between vertices v_i and v_j, n being the number of nodes.

Traditional deep learning has majorly focused on the euclidean-space. In the deep learning era, one of the major works is CNN which utilizes the shift-invariance, local connectivity, and compositionality of image data. However, numerous applications, ranging from interactions and activities on social media, molecular activity and structure in drugs and program verification are based complex graph based data. Designing machine learning algorithms on such data is difficult as the graph domain is irregular i.e. each graph has a variable size of unordered nodes, each of which may be connected to a different set of other nodes. Furthermore these nodes are not independent of each other and are related via edges (which contains some linkage information). These properties makes it hard for defining a convolution operator in this domain [12].

As per recent works the graph networks can be categorized under the following categories:

- Graph Recurrent Networks: These networks generalize the concept of recurrent networks in graphs but instead of learning sequence outputs these networks outputs are node representations, graph representations, etc which can be further used for classification or detection tasks. The core idea in these networks are (i) node annotations (ii) information propagation model (which can use gated equivalents of LSTM) and (iii) output models (which outputs the node/graph representations or selects specific nodes [80, 58].

- Graph Convolution Networks(GCN): These networks mimick the convolution operators from traditional data like images or grids to graph data. Specifically, they learn a function which aggregates a node and its neighbors' features [13, 21, 48, 56, 30, 75, 22, 16].

- Graph Attention Networks: Similar to GCN, these networks learn a function which aggregates a node and its neighbors features [93, 106]. The major difference with GCN is the use of parametric weights to implicitly capture the edge relationship/importance.

- Graph Auto-Encoders: To learn graph representations graph auto encoders employ an unsupervised approach. They consist of an encoder (underlying network could be GCN) and a decoder which utilizes a link-decoder prediction [76, 104].

- Graph Generative Networks: Synthesizing complex chemical compounds with requisite physical and chemical properties is a challenging task. Graph generative networks have shown promise in this highly specialized domain by alternatively generating nodes and edges and employing adversarial training [59, 11].

GCN can be further divided among two sub-categories: spectral-based and spatial-based. In spatial based setup, GCN learns filters which aggregates features of a node and its neighbors. In spectral based setup, the learned filters are defined from the perspective of graph signal processing [12] which work in a graphs' Fourier basis and manipulate/alter the magnitude of the frequencies. We describe this in detail in Subsection 2.3.2.

2.3.1 Spectral Graph Theory

For graphs the underlying non-euclidean topology is analyzed in terms of eigenvalue-eigenvector spectrum of graph in the field of *spectral graph theory* [19]. Spectral analysis of a graph starts first by constructing its Laplacian matrix $\boldsymbol{L}_{n\times n}$. If $\boldsymbol{D}_{n\times n}$ is the degree matrix of the graph with its diagonal entries $d_{i,i}$ representing the degree of node i, then $\boldsymbol{L}$ is defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$.

Laplacian matrix represents the second-order derivative operator on the functions on a graph. In this respect it captures the topology of the graph and becomes a basis for defining all the operators over the graph. Laplacian matrix provides a link between discrete graph representations and vectors in continuous Euclidean space. Mathematical operators defined in the continuous space are extended in the discrete space via the graph Laplacian matrix. Continuous-space kernels operate on vector valued functions defined by physical processes in that space. The discrete counterparts of these operators are the kernels defined as functions of the graph Laplacian. These kernels operate on functions or signals defined on the nodes of the graph.

One of the important properties of Laplacian matrix is that it is a positive semi-definite matrix, hence it is eigen-decomposable:

$$\boldsymbol{L} = \boldsymbol{U}_{n*n}\Lambda_{n*n}\boldsymbol{U}^T_{n*n} \tag{2.1}$$

where Λ is a diagonal matrix and U consists of orthonormal column vectors. Hence this can also be written as

$$\boldsymbol{L} = \sum_{i}^{n} u_i\lambda_i u_i^T \tag{2.2}$$

where u_i is a eigen(column)-vector and λ_i is the corresponding eigenvalue.

An important random process over a graph is diffusion. These graph kernels operate on the real valued functions $f : V \to \mathbf{R}$ on the set V of the graph. Diffusion of a function on a graph captures the spatial extent of spread of the function with time. The kernel that captures the spatio-temporal extent of diffusion is the diffusion kernel H. The scale dependent diffusion kernel is:

$$\boldsymbol{H}(t) = \boldsymbol{U}e^{-\lambda t}\boldsymbol{U}^T = \sum_{i}^{n} u_i e^{-\lambda_i t}u_i^T \tag{2.3}$$

The exponential function of the graph Laplacian is the kernel over the graph that represents the extent of diffusion of the function f at the *diffusion scale* t. Diffusion (or heat) kernel is uniquely defined at its scale given the graph.

All these kernels can be principally put in another framework also. This framework extends the concepts of signals and systems onto irregular domains such as graphs.

2.3.2 Graph Signal Processing

In the signal processing domain, a function is treated as a signal. If every node of the graph contains a signal(real-value), the combined signal over the entire graph is called a graph-signal $\boldsymbol{s} \in \mathbb{R}^n$. This

graph-signal forms a pattern over the graph-topology. The emerging field of *graph-signal-processing* analyzes such graph-signals in terms of their components on defined *harmonics* based on the algebraic and spectral properties of graph-topologies [85].

We will first conceptualize the notion of frequency over a graph and the corresponding Fourier basis defining the transform. For a 1-dimensional signal $s(t)$ varying with time t, the Fourier transform is as follows:

$$\mathcal{U}(\omega) =< s(t), e^{j\omega t} >= \int_{\mathbb{R}} s(t)e^{-j\omega t}dt, \tag{2.4}$$

where ω is the angular frequency of the complex exponential basis signal and $j = \sqrt{-1}$. Every component of the signal on a Fourier basis signal is the inner product between the two. Consider the eigenfunctions of the 1-dimensional Laplacian operator:

$$\Delta(e^{j\omega t}) = \frac{\partial}{\partial t}\Big(\frac{\partial}{\partial t}(e^{j\omega t})\Big) = j\omega\frac{\partial}{\partial t}(e^{j\omega t}) = -\omega^2 e^{j\omega t}. \tag{2.5}$$

These eigenfunctions are the Fourier basis signals with their corresponding eigenvalues as the square of the frequency. Thus the graph Fourier transform is defined as the expansion of a graph-signal in terms of the eigenvectors of graph-Laplacian:

$$\hat{\mathbf{s}}_i =< \mathbf{s}, u_i >= \sum_{k=1}^{n} \mathbf{s}_k u_i^*(k)\hat{\mathbf{s}}_{n\times 1} = \boldsymbol{U}^\top_{n\times n}\mathbf{s}_{n\times 1}(since \mathbf{L}\mathbf{u}_i = \lambda_i \mathbf{u}_i) \tag{2.6}$$

where $\boldsymbol{U}^\top$ representing the conjugate of the eigenvectors and $\boldsymbol{\Lambda}$ representing the square of the frequencies of the graph-topology.

In the classical Fourier analysis the angular frequencies ω carry a specific meaning of frequency. The smaller the ω the slower the oscillation of the Fourier basis signal and vice-versa. And for $\omega = 0$ the basis signal takes a constant value. Similarly in the case of graphs, the graph Laplacian eigenvectors corresponding to lower (higher) λ_i's vary slowly (rapidly) on the graph, introducing the concept of graph harmonics. If there is a strong edge between two nodes, the dissimilarity between the values of the eigenvector on the two locations will increase with λ.

The inverse graph Fourier transform is also easily deducible:

$$\begin{aligned} s(t) &=< \mathcal{U}(\omega), e^{-j\omega t} > \\ \boldsymbol{s}(t) &= \boldsymbol{U}\hat{\boldsymbol{s}}. \end{aligned} \tag{2.7}$$

In the above equation, the graph-signal $\boldsymbol{u}$ is represented as a linear combination of graph harmonics (columns of $\boldsymbol{U}$) where elements of $\hat{\boldsymbol{s}}$ form the coefficients of linear combination which represent contribution of each component graph harmonic.

One of the most important aspects of signal processing is filtering of signals which have applications ranging from noise-removal, compression, communication etc. Filtering operations in spatial domain require convolution operations. To define a convolution operation a notion of regular neighborhood of each point in the given space is necessary. Such a notion of neighborhood regular neighborhood could

be easily found out in euclidean spaces(or regular-grid spaces) such as temporal signal(audio), spatial signal(image) or spatio-temporal(video). In these domain the convolution operator could be easily define by sliding a filter of fixed size in a pre-defined method(example left to right and then top to bottom). Hence convolution can be simply computed as an integral/summation at each point with the center of the filter colliding with the current point in space.

Following equation is example of convolution of a signal g and a filter h, both defined in temporal-domain.

$$(f \star g)(t) = \int_{-\infty}^{\infty} f(t) * g(\tau - t) d\tau \tag{2.8}$$

Here, the symbol $\star$ defines the convolution operation.. Frequency filtering of signals preserves this notion of convolution by modulating the contribution coefficients of the component harmonics. This implies that the convolution operation behaves as a pointwise multiplication operator in the Fourier domain. Hence if the signal(and the filter) were first converted into their Fourier basis, the convolution operation would be reduced to a Hadamard product of Fourier domain signal. This can be represented as:

$$\hat{\boldsymbol{y}} = \hat{\boldsymbol{h}} \odot \hat{\boldsymbol{s}}. \tag{2.9}$$

where $\boldsymbol{h}$ and $\boldsymbol{s}$ are the Fourier representation of the filter and signal respectively. The operator $\odot$ defines Hadamard product / element-wise product of two vectors.

Defining a spatial filter in a graph is non-trivial as the neighborhood is irregular i.e. neighborhood at each vertex $v \in V$ is not constant. Hence Signal Processing in Graphs uses Fourier domain to define convolution operator on graphs. The above operation 2.9 can be generalized by considering spectral domain signals as functions over the eigenvalues of the graph, which carry the notion of frequency of the component graph harmonics. Specifically in case of graphs this becomes:

$$\hat{y}(\boldsymbol{\Lambda}) = \hat{h}(\boldsymbol{\Lambda})\hat{s}(\boldsymbol{\Lambda}). \tag{2.10}$$

Here $\hat{\boldsymbol{s}}$, $\hat{\boldsymbol{h}}$ and $\hat{\boldsymbol{y}}$ are the graph signal, filter and the filtered output defined in spectral domain respectively.

We can take an inverse graph Fourier transform of $\hat{y}$ to get the final filtering output in vertex domain:

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{U}\hat{y}(\boldsymbol{\Lambda}) \\ &= \boldsymbol{U}\left(\hat{h}(\boldsymbol{\Lambda})\hat{s}(\boldsymbol{\Lambda})\right) \\ &= \boldsymbol{U}\hat{h}(\boldsymbol{\Lambda})\boldsymbol{U}^\top \boldsymbol{s} \\ &= \hat{h}(\boldsymbol{L})\boldsymbol{s} \end{aligned} \tag{2.11}$$

The graph filtering approach can be used to conduct various techniques such as smoothing, translation, diffusion etc. As a special case, if $\hat{h}$ is considered to be an exponential function, we get a diffusion kernel or heat kernel over the graph Laplacian as defined in Equation 2.3.

Chapter *3*

Image Annotations with Multiple Kernel Learning(MKL)

The task of automatic image annotation attempts to predict a set of semantic labels for an image. Majority of the existing methods discover a common latent space that combines content and semantic image similarity using the metric learning kind of global learning framework. This limits their applicability to large datasets. On the other hand, there are few methods which entirely focus on learning a local latent space for every test image. However, they completely ignore the global structure of the data. In this work, we propose a novel image annotation method which attempts to combine best of both local and global learning methods. We introduce the notion of neighborhood-types based on the hypothesis that similar images in content/feature space should also have overlapping neighborhoods. We also use graph diffusion as a mechanism for label transfer.

The basic assumption in image annotation is that the visual content of an image captures a wide variety of semantics at different levels of granularity. Additionally, the label co-occurrence patterns also model the semantic similarity between images. For example, the possibility of a car and a $road$ tag appearing together in an image is highly likely as compared to a car and a $plane$. Therefore, existing methods have tried to model label-to-label [65], image-to-image [34] and image-to-label [14] similarities or a combination of them [97, 63].

In context of image-to-image and image-to-label similarities Nearest Neighbor (NN) based approaches have been largely successful and intuitive for image annotation. Recent methods either employ a global (metric) learning technique [96, 34] or a local query specific model [45] for addressing the class-imbalance problem. However, while the former suffers from the problem of scalability due to global metric learning bottleneck, the latter fails to capture the global latent structure of the data as it is too focused on query specific neighborhood structure. An alternate approach in [73] addresses the class-imbalance by performing scale-dependent label diffusion on global hypergraph in a transductive setup. However, their method also suffers from the scalability issue (due to SVD decomposition of large dense matrices). Many recent deep learning methods also propose to learn end-to-end network for solving image annotation task [72, 98, 62].

In this paper, we focus on bridging the gap between purely global and local modeling of the image annotation task. The key hypothesis is that similar images in feature space also have similar labels,

hence two vicinal images in feature space should also have overlapping neighborhoods. Each of these neighborhoods (corresponding to an image) can be statistically characterized by constructing the label histogram of all their associated images. We refer to these label histogram features as Local Label Distribution (LLD) features. Thus, two similar images should have similar LLD feature representation which represent similarity in neighborhood. Hence we propose to learn a local label-transfer model for each such neighborhood-type (cluster) separately. This characterization of images by neighborhood-types also inherently captures the global latent structure of data.

Subsequently, each local model is formulated as Multiple Kernel Learning (MKL) task, using a family of multi-scale diffusion kernels. The MKL formulation minimizes the sum of squared error between the ground truth labels (known for each training image) and the labels predicted with multi-scale diffusion over the associated local graph. Such diffusion is performed by linearly combining a set of scale dependent diffusion kernels. A closed form solution exists for obtaining the optimal kernel combination coefficients (parameters of local model). Thus, MKL parameters per neighborhood-type are learnt over the training data. At test time, we construct and map the neighborhood structure of each query image to an existing neighborhood-type to retrieve the best parameter of local model. Finally, we construct the local graph for this query image and diffuse the label using these parameters for subsequent prediction.

3.0.1 Our Contributions

In this chapter we contribut in the following ways:

- We propose a new label histogram characterization (LLD features) of the image neighborhood enabling us to discover the neighborhood-types in the dataset.
- We propose a MKL formulation as local learning model and derived a closed form solution for obtaining the model parameters.
- We propose a diffusion scale normalization procedure for effectively combining diffusion over multiple graphs.

3.1 Proposed Approach

This section will provide a detailed description of each individual module in the proposed train and test phase flow pipeline depicted in Figure 3.1. Let $\mathbf{X}^{tr} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ be the feature-vector representation of training set of images with corresponding known ground truth labels $\mathbf{Z}^{tr} = \{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$, where each $\mathbf{z}_i$ is a binary vector of size l denoting presence/absence of labels in the image $\mathbf{x}_i \in \mathbf{X}^{tr}$.

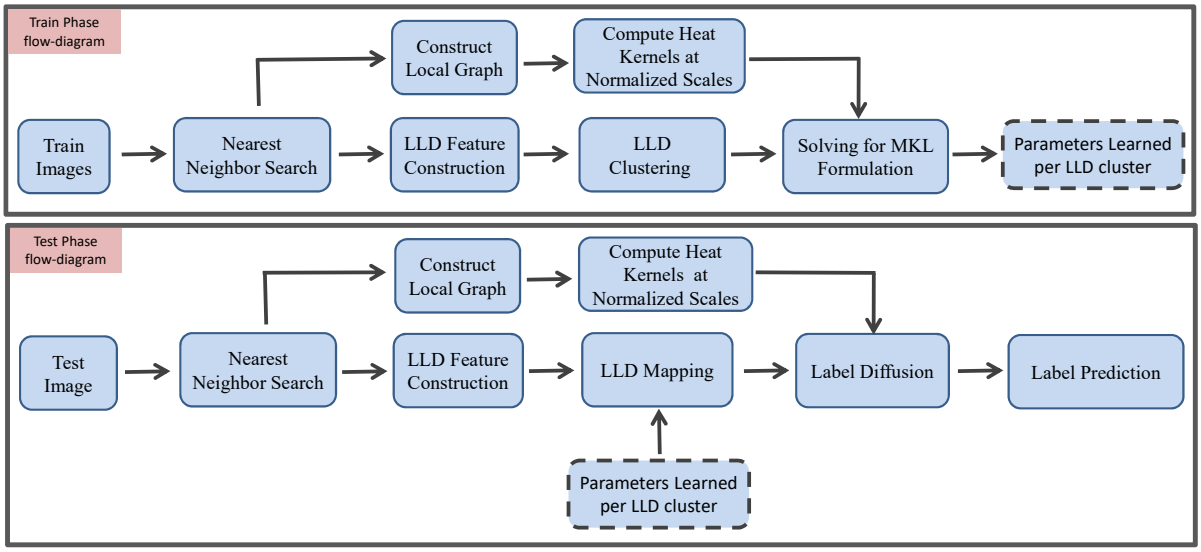


Figure 3.1: Pipeline showing flow of testing & training phase

3.1.1 Nearest Neighbors Search

This module performs feature space NN search for a given image in order to discover a group of similar images. Instead of performing a global search, we opt for quantizing the feature space image representation by employing the paralleizable k-means clustering algorithm [1] on training data and finding a fixed g number of clusters in an offline manner. From all these clusters, we subsequently find the top η closest cluster centers in feature space, then we perform the local NN search in those clusters by retrieving a fixed δ number of similar images from each of them. All such retrieved images form the neighbourhood of a given image. We denote this set of nearest neighbor images obtained with this method for a given image $\mathbf{x}$ as $\mathcal{N}(\mathbf{x})$. Note that $|\mathcal{N}(\mathbf{x})| \leq \eta\delta$, as a cluster can have less than δ images. This naturally provides diversity and scalability over the exhaustive NN search.

3.1.2 Local Graph Construction

This module constructs an undirected weighted graph $\mathcal{G}(V, E, \mathcal{W})$ for an image $\mathbf{x}$ using the neighborhood $\mathcal{N}(\mathbf{x})$, where the input image and each of the selected images in $\mathcal{N}(\mathbf{x})$ images become nodes of the graph.

We construct a local graph by connecting each node to its k most similar nodes using an inverse euclidean distance similarity function over the corresponding image features. We use the standard Gaussian kernel over the feature space as the similarity function, i.e., $Sim(x_1, x_2) = \exp(-||x_1, x_2||_{l2}/\sigma^2)$ to assign weights to these edges.

Note that here $|V| = \zeta + 1 \leq \eta\delta + 1$ where $|\mathcal{N}(\mathbf{x})| = \zeta$

[1] https://github.com/serban/kmeans

3.1.3 LLD Feature Construction, Clustering and Mapping

This module first constructs an l-dimensional histogram feature $\mathcal{F}(\mathbf{x})$ for each image representing the LLD of a given image $\mathbf{x}$, by taking the sum of ground truth labels of all the samples in $\mathcal{N}(\mathbf{x})$:

$$\mathcal{F}(\mathbf{x}) = \sum_{\forall \mathbf{x_i} \in \mathcal{N}(\mathbf{x})} \mathbf{z}_i. \tag{3.1}$$

We employ parallelizable k-means clustering algorithm over the LLD features corresponding to all training images to get a fixed number of cluster-centers (c) which act as representatives of the neighborhood-types. To map an image to a neighborhood-type we just need to compute its closest neighborhood-type (cluster-centers) in this l-dimensional space. As discussed in Section 3, we discover the clusters in LLD space and learn a local model per cluster.

3.1.4 Diffusion Kernel

In this module we construct a family of diffusion kernels at different diffusion scales for a given local graph computed in the previous module. In each weighted graph $\mathcal{G}(V, E, \mathcal{W})$, training images with labels acts as heat sources that are diffused/propagated to all the other nodes in the graph where we aggregate the information and subsequently use for prediction.

We have introduced graph laplacian in Subsection 2.3.1 (Mathematically in Equation 2.1). Diffusion kernel is a positive semi-definite, non-linear family of kernels and can be used for defining distances over non-euclidean spaces and for multiscale-multilabel-diffusion in graphs. It has been used for multi-scale label diffusion over graphs [90], and a variety of other applications 3D Shape Matching [84] and Robotics graphSLAM [20] Subsequently, the scale dependent diffusion kernel matrix is defined as: $H(t) = Ue^{-\Lambda t}U^T$ where $t > 0$ is the parameter of the diffusion. Every entry $H(i, j, t)$ of the diffusion kernel can be interpreted as the amount of heat diffused from node v_j to node v_i at scale t while considering v_j as a point heat source of unit magnitude. We use the diffusion kernel matrix at m different scales to diffuse each label. This is different from [73] as we use multiple scales for all labels rather than using specific scales for different frequency type of labels.

3.1.4.1 Diffusion Scale Normalization.

Instead of manually choosing diffusion scales for all local graphs, we propose to find a set of normalized scales per graph using the structure of the local graph. This structure is captured by the spectrum (eigenvalues) of the graph. Such a normalization is very important as the diffusion scale value is relative to the graph structure/topology. Interestingly, if we see the plot of exponential function

$$f(\lambda, t) = e^{-\lambda t} = \theta \tag{3.2}$$

in Figure 3.2, we see that one can find the normalized values of the diffusion scale parameter t (varying from smaller to larger values) by fixing the values of θ and index of λ.

$$t = -log(\theta)/\lambda \tag{3.3}$$

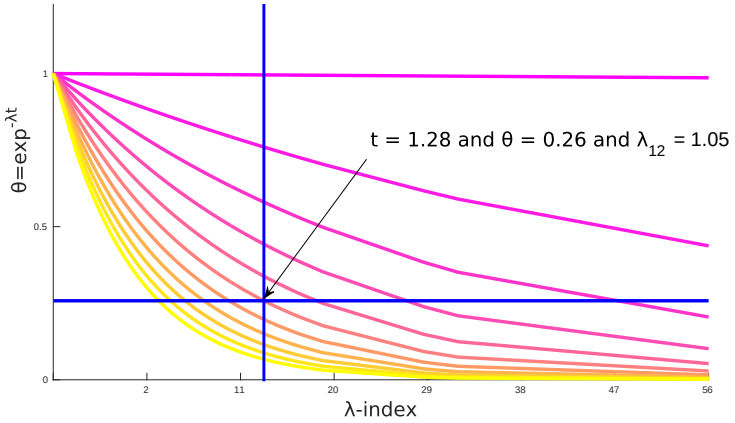


Figure 3.2: Diffusion scale normalization on a sample graph of 56 nodes. In this case we use the 12th ($= 0.2 * |56|$) eigenvalue and θ = 0.26 for computing the scale normalized t value. In general a set of θ values will be chosen for defining bank of diffusion kernels

3.1.5 MKL Formulation for Label Diffusion

In this section, we outline our MKL formulation for learning the label diffusion parameters locally for each LLD cluster/neighborhood type.

Let $\mathbf{X}_c^{tr} \subset \mathbf{X}^{tr}$ be the subset of n_c training images and $\mathbf{Z}_c^{tr} \subset \mathbf{Z}^{tr}$ be the ground truth labels in c-th LLD cluster. We can write $\mathbf{X}_c^{tr} = [\mathbf{x}_1, \cdots, \mathbf{x}_{n_c}]$ and $\mathbf{Z}_c^{tr} = [\mathbf{z}_1, \cdots, \mathbf{z}_{n_c}]$.

For each image $\mathbf{x}_k \in X_{tr}^c$ there is a local graph $\mathcal{G}_k$ (Section 3.1.2) with $\zeta_k + 1$ nodes where the node with the last index is $\mathbf{x}_k$ itself. Let $\mathbf{z}_k$ be the ground truth label of $\mathbf{x}_k$ and $\mathbf{Y}_k = [\mathbf{y}_1 \cdots \mathbf{y}_l]$ be the transpose matrix of ground truth labels (i.e., $[\mathbf{z}_1 \cdots \mathbf{z}_k]^T$) for all the other ζ_k images (nodes) in the local graph appended with a $\mathbf{0}$ row vector representing labels for $\mathbf{x_k}$ itself. Note that $\mathbf{y}_i$ is a column vector of dimension $(\zeta_k + 1) \times 1$. The last row is appended for compatibility in matrix multiplication.

Let $\tau = [t_1, \cdots, t_m]^T$ be the set of m normalized diffusion scale parameters used for defining the diffusion kernels: $H_k(t_1), \cdots, H_k(t_m)$ (Section 3.1.4.1). Here each $H_k(t_i)$ is a $(\zeta_k + 1) \times (\zeta_k + 1)$ dimensional matrix. Let $\mathbf{e} = [0, \cdots, 0, 1]$ be a $(\zeta_k + 1) \times 1$ dimensional vector, then $h_k^i = e^T H_k(t_i)$ represents the last row of the (symmetric) diffusion kernel matrix.

It is important to note that since the training image $\mathbf{x}_k$ is kept at the last position in the index order in graph $\mathcal{G}_k$, only the last row of $H_k(t_i)$ (i.e., h_k^i) is sufficient to perform label diffusion (at scale t_i) from all other images (nodes) in the graph.

Since we know the ground truth labels for image $\mathbf{x}_k$, and if we take β_j^c to represent the diffusion contributions of the diffusion kernels $h_k^i \forall i \in \{1, \cdots, m\}$ for j^{th} label in the c^{th} cluster, we can obtain an MKL formulation as a minimization criterion:

$$\min_{\beta_c} \sum_{k=1}^{n_c} ||\mathbf{\Gamma}_k \beta_c - \mathbf{z}_k||^2, \tag{3.4}$$

where,

$$\beta_c^j = [\beta_c^{j1}, \beta_c^{j2}, ..\beta_c^{jm}]_{(1\times m)} \tag{3.5}$$

$$\beta_c = [\beta_c^1, \beta_c^2, ..\beta_c^l]_{(lm\times 1)}^T \tag{3.6}$$

$$\Gamma_k = \begin{bmatrix} (\mathbf{M}_k\mathbf{y}_1)^T & & & \\ & \ddots & & \\ & & (\mathbf{M}_k\mathbf{y}_r)^T & \\ & & & \ddots \\ & & & (\mathbf{M}_k\mathbf{y}_l)^T \end{bmatrix}_{(l\times lm)} \tag{3.7}$$

and

$$\mathbf{M}_k = [h_k^{1T}, h_k^{2T}, \ldots, h_k^{mT}]_{(m\times(\zeta_k+1))}^T. \tag{3.8}$$

By combining Eq. 3.4, 3.6, 3.7 with simple algebraic manipulations, we can write the simplified MKL formulation as:

$$\min_{\beta_c} ||\hat{\mathbf{\Gamma}}_c\beta_c - \mathcal{Z}_c||^2, \tag{3.9}$$

where,

$$\hat{\mathbf{\Gamma}}_c = [\mathbf{\Gamma}_1^T, \cdots, \mathbf{\Gamma}_{n_c}^T]^T \tag{3.10}$$

is a $(n_c l \times lm)$ matrix and

$$\mathcal{Z}_c = [z_1^T, \cdots, z_{n_c}^T]^T \tag{3.11}$$

is a $(n_c l \times 1)$ vector.

The minimization of proposed MKL formulation Eq. 3.9 can be achieved by finding the optimal value of parameter β_c in a closed form manner as:

$$\beta_c = (\hat{\mathbf{\Gamma}}_c^T\hat{\mathbf{\Gamma}}_c + \epsilon I)^{-1}\hat{\mathbf{\Gamma}}_c^T\mathcal{Z}_c, \tag{3.12}$$

It is important to note that the $\hat{\mathbf{\Gamma}}_c$ is a very sparse and low rank matrix. This sparse structure can be useful in efficiently computing its singular value decomposition and hence the pseudoinverse of the $\hat{\mathbf{\Gamma}}_\mathbf{c}$ which is relatively inexpensive as KCCA or hypergraph Laplacian eigenvector-decomposition.

Since our method solves MKL at cluster level in a closed form manner, it is scalable to large data.

3.1.6 Label Diffusion & Prediction

For a test image $\mathbf{x_q}$, $\mathcal{F}(x_q)$ is used to find s nearest LLD neighborhood-types represented as $\mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_s]$. The associated pre-learned MKL parameters $[\beta_1, \cdots, \beta_s]$ for the selected clusters are used for independent diffusion over the local graph $\mathcal{G}_q$ on $\mathbf{x}_q$. This is achieved by first computing the respective

Table 3.1: Dataset details: Row 2-4 contain basic dataset information, Row 5-6 denote the statics in the order- median, mean and max.

Dataset Information	PascalVOC-2007 [26]	MIRFlickr-25k [41]
Total Number of Images	9963	25000
Vocabulary Size	20	38
Train/Test Split	5011/4092	12500/12500 [92]
Labels/Images	1, 1.51, 6	5, 4.7, 17
Images/Labels	257.5, 379.2, 2050	995.5, 1560, 5216

$\mathbf{\Gamma}_q$ matrix sized $(l \times lm)$ and then multiplying it with the pre-learned parameter vector β_c sized $(lm \times 1)$. Finally, we take an average of these diffused values. Label diffusion is performed as:

$$\mathbf{z}_q^{\text{diffused}} = \frac{1}{s} \sum_{c=1}^{s} \mathbf{\Gamma}_q \beta_c \tag{3.13}$$

where $\mathbf{z}_q^{\text{diffused}}$ is the $(l \times 1)$ vector of the diffused labels. Finally, a fixed set of r labels is predicted for $\mathbf{x}_q$ by choosing labels corresponding to top r values in $\mathbf{z}_q^{\text{diffused}}$.

Explicitly diffusing labels at multiple normalized scales handles the prevalent class imbalance. We also emphasize the idea of diffusion as a method of indirect influence. While a neighbor may not be connected to the query image but it could still influence the query image through other nodes(images) via diffusion.

3.2 Datasets

We describe the datasets used in this chapter in the following subsections. Table 3.1 summarizes basis information on these datasets.

3.2.1 PascalVOC-2007

PASCAL Visual Object Classes (VOC) challenge [26] datasets are widely used as the benchmark for multi-label classification and detection. In this work we use this dataset for the task of multi-label classification. The VOC 2007 dataset contains 9963 images. We follow the train/test split from [98] and obtain 5011 training images and 4952 test images. VOC consists of 20 semantic concepts which are $plane$, $bike$, $bird$, $boat$, $bottle$, bus, car, cat, $chair$, $table$, cow, dog, $horse$, $motor$, $person$, $plant$, $sheep$, $sofa$, $train$ and tv.

3.2.2 MIRFlickr-25k

This dataset contains images downloaded from Flickr and was introduced in [41] for evaluating keyword-based image retrieval. It consists of 25000 images and we follow an equal split of train and test (12500 images each) as used in [92]. Metadata, GPS and EXIF information are also provided in the dataset but we do not use any of these in our method.

The dataset is annotated with 38 semantic concepts. These 38 tags are labeled into two parts: user tags (a set of 20 labels) which are labeled based on a "wide sense" of the word and expert tags (a set of 18 tags, repeated from 20 tags) which are labeled based on "narrow sense" of the word. An example for "wide sense" is labeling an image as car when the snapshot seems to be taken from a car, while the snapshot itself doesn't contain a car. The user tags are noisy, weak and overly personalized but are present in abundance at an average of 8.94 tags per image. The expert tags are accurate but are scarce, tagged at an average of 2.78 per image. The tags are $animals$, $baby$, $baby\_r1$, $bird$, $bird\_r1$, car, car_r1, $clouds$, $clouds\_r1$, dog, dog_r1, $female$, $female\_r1$, $flower$, $flower\_r1$, $food$, $indoor$, $lake$, $male$, $male\_r1$, $night$, $night\_r1$, $people$, $people\_r1$, $plant$, $portrait$, $portrait\_r1$, $river$, $river\_r1$, sea, sea_r1, sky, $structure$, $sunset$, $transport$, $tree$, $tree\_r1$ and $water$. Note that the suffix $r1$ denotes expert labels. Additionally we also observe that 419 images in the dataset do not have any of the 38 semantic label annotations

3.3 Experiments & Results

3.3.1 Features and Evaluation Method

Deep-learning based features have proven to be effective in image representation [73, 96] and hence we use outputs from *fc7-layer* of VGG16 network (pretrained on ImageNet) [86] to represent an image. We pre-compute these features to avoid re-computations for ablation studies or parameter changes.

To analyze the annotation performance, we consider precision, recall, F1-score, average precision (AP) and mean average precision (mAP). We predict a fixed number of r labels per image which is set to be the mean number of labels per image in the dataset. Let a label w_i be present in $m1$ images as ground-truth and is predicted for $m2$ images where $m3$ of them are correct. The precision for w_i is $m3/m2$ and recall is $m3/m1$. Mean precision (P) and recall (R) is the precision and recall values averaged over all the labels. $F1$ measure is the harmonic mean of P and R. We also report AP and mAP by evaluating ranking of all the images.

3.3.2 Experiments

We set $k = 6$ in kNN graph construction in section 3.1.2 and number of normalized diffusion scale parameters(m) as 100. Additionally we set $s = 3$ for PascalVOC-2007 and $s = 5$ for MIRFlickr-25K.

Table 3.2: Comparison of popular methods on different evaluation metrics for MIRFlickr-25k Dataset for $\mathbf{r = 5}$

Method	$P@r$	$R@r$	$F1@r$	mAP
SVM [92]	-	-	-	52.3
TagRel [57]	41.5	72.1	52.7	68.9
TagProp [65]	45.5	70.1	55.2	70.8
2PKNN [96]	46.4	70.9	56.1	66.5
SVM [94]	38.8	72.4	50.5	72.7
HHD [73]	-	-	-	75.0
Our Method	51.0	59.9	55.1	66.3

We observed that the performance variation due to change in m from 32 to 100 was meager amounting to less than 1%.

In the diffusion scale normalization(Section 3.1.4.1), θ is chosen as m equally spaced values between 0.001 to 1.0 (corresponding t will vary from large to small scales of diffusion) and the index of the eigenvalue is chosen as the closest integer value greater than $0.2 \times |V|$. This ensures that a wide variety of scales are used in diffusion.

We find the values for the rest of the hyper-parameters via cross-validation by dividing the train dataset into two parts ($5:1$ ratio, treating the smaller part as validation set) while maximizing $F1$. The best performing parameters on validation set were used to evaluate performance on the test data. For η and δ we explore from the following set $\{5, 12, 20, 28\}$ to find the best performing values. We also vary the number of cluster centers in LLD (c) and the number of clusters in image-feature space clustering (g). The best performing values for MIRFlickr-25k were found to be $\eta = 5$, $\delta = 12$, $c = 100$ and $g = 30$ and for PascalVOC-2007 were found to be $\eta = 5$, $\delta = 20$, $c = 45$ and $g = 20$.

3.3.3 Results and Discussions

Table 3.2 shows the performance comparison of the proposed method with existing methods that uses VGG16 features. The obvious understanding one can make here is that there is non-agreement between F1 and mAP measures. The mAP considers the global ranking of all images corresponding to each label instead of just considering top r labels for computation of average precision.

We can see that our method performs very close to the state of the art 2PKNN method and also has similar mAP. This small disparity in performance can be attributed to the fact that our method does not consider KCCA and metric learning type of fully global operations.

Figure 3.3 shows the distribution of both ground-truth and predicted label frequency in test images on two datasets. For MIRFlickr-25k, a large section of predicted label frequency curve (including low and high frequency labels) closely overlaps with that of ground truth. However, the medium frequency labels (in the middle) are over-predicted at the cost of suppression of few frequent occurring labels (right

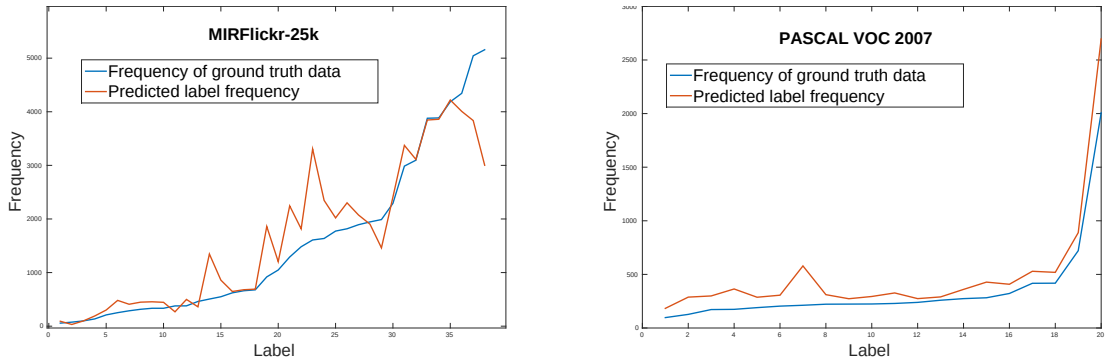
Table 3.3: Label specific (Average Precision in %) for all labels, mAP, P@r, R@r and F1@r with $\mathbf{r = 2}$ on PascalVOC-2007 dataset.

	CNN-RNN [98]	Our Method
plane	96.7	92.8
bike	83.1	84.7
bird	94.2	91.3
boat	92.8	81.7
bottle	61.2	41.3
bus	82.1	83.9
car	89.1	89.0
cat	94.2	86.3
chair	64.2	55.7
table	70.0	68.4

	CNN-RNN [98]	Our Method
cow	83.6	71.9
dog	92.4	86.7
horse	91.7	89.4
motor	84.2	82.7
person	93.7	91.8
plant	59.8	54.0
sheep	93.2	75.1
sofa	75.3	57.1
train	99.7	92.6
tv	78.6	66.9

	CNN-RNN [98]	Our Method
P@r	-	53.8
R@r	-	77.7
F1@r	-	63.6
mAP	84.0	77.2

Figure 3.3: Distribution of label frequency in MIRFlickr-25k and PascalVOC-2007 test images.



tail). This is prevalent due to the nature of our approach, but we accept this as a trade-off in order to address the issue of class imbalance by accurately predicting lower frequency labels in images. In case of PascalVOC-2007, we observe our curve running parallel to the one of ground truth but with a shift of a few units. This shift is due to the difference that on an average only 1.5 labels are associated with image in the ground truth annotation but we predict 2 labels per image.

Table 3.3 reports the label specific AP and mAP results along with P, R and F1 measure at $r = 2$ for PascalVOC-2007 dataset. Except a few categories like $bike, bus$, the performance of our method is below the state of the art but performs decently with respect to the state of the art Deep Learning based methods in [98] terms of mAP measure. The F1 measure is 63.6 which we believe is good. However, we were unable to find any other existing paper reported F1 measure on this dataset. Hence we could not compare our performance with [98] in terms of F1 measure, which we believe is more relevant to the task at hand.

Figure 3.4 shows the qualitative results on a few examples from the MIRFlickr-25k dataset. Labels in red-color denote tags which are not present in the ground-truth of the image but are semantically meaningful. This indicates towards the incomplete-labeling in the dataset. Row1 depicts images which consists of frequent labels in the ground-truth($male$, $people$) while Row2 consists of images with no ground-truth and Row3 contains images with rare labels ($baby$, $portrait$) in the ground-truth. From Row1 and Row3 we observe that our method performs well on the frequent and rare labels. We also observe that for images with no ground-truth (Row2) the predictions are semantically relevant to the image-content. As mentioned in Section 3.2.2 we know that tags with suffix $r1$ are clean and belong to $expert$ labels. From the qualitative results we observe that our model has correctly predicted these $expert$ tags. Similarly noisy tags and incorrect tags(as in Row 2) may also result in poor quantitative performance.

Figure 3.5 shows qualitative results on a few examples from the PascalVOC-2007 dataset. Row 1 depicts images consisting of frequent labels in the ground-truth($person$, car) while Row 2 depicts images consisting of rare labels in the ground-truth($boat$, cow). We observe that our model performs decently on rare labels.

Figure 3.4: Qualitative results for label prediction on MIRFlickr-25k dataset. Row1: images with frequent labels($male$, $people$); Row2: images with no ground truth label given; Row3: images with rare labels($baby$, $potrait$). Green: Labels present in ground-truth and our model's predictions (true positives) , Blue: Incorrect predictions by the model and Red: Labels are not present in ground truth but are semantically meaningful. Red and blue labels combined form False Positives

.

Figure 3.5: Qualitative results for label prediction on PascalVOC-2007 dataset. Row1: images with frequent labels($person$, car); Row2: images with rare labels($boat$, cow). Green: Labels present in ground-truth and our model's predictions (true positives) , Blue: Incorrect predictions by the model

Chapter *4*

Image Annotations with Graph Deep Learning

In the recent era of deep learning, image annotation models have focused on three modalities:

- **Image Representation**: Regular grid CNNs have shown tremendous progress in various tasks such as image segmentation, single label annotation and it has been shown that these features can be used off-the shelf. This stage is thus focused on finding the best representation for an image. Usually pre-trained VGG/Resnet features are utilized in this stage for annotation tasks as these models are trained for (single-label) annotation tasks. Recent works have also focused on fine-tuning this stage to make the model end-to-end and to improve on metrics.
- **Label Modelling**: This step involves modelling the interdependence and patterns between the labels. Labels or words are highly correlated but these relations are non-trivial and needs to be modelled. Biases may arise due to the data annotation techniques as well as due to the underlying data distribution(eg. set of natural images will have bias towards labels like trees, mountains, seas)
- **Joint Space and Prediction**: This stage focuses on bringing the representation from the first two stages together for prediction. An effective way of combining the two modalities may result in better understanding and result in improved performance, for eg. [62] showed generic improvements on CNN-RNN models.

Our focus in Chapter 3 was bringing the best of global and local methods in the domain of image annotation. In this work we continue to do so while moving to the realm of graph deep learning. Since graph deep networks used independently aren't at par with traditional deep learning approaches (such as VGG, ResNet) even on toy datasets like MNIST, we only model label dependence/co-relations with them. In deep learning methods, each sample is treated separately both in training and testing phases. This property categorizes deep learning methods([99, 62]) under local methods. However, our proposed approach requires all of training data to populate an adjacency graph which is the core of graph deep learning methods. Hence our approach tries to utilize the best of global and local approaches. Specifically, we model label co-relations and intrinsically capture the diffusion of information through them via the use of graph convolutions. We also investigate weather graph convolution network (as formulated by [21]) can be extended to improve the performance in image annotation task.

As discussed in Section 2.3 a plethora of methods have surfaced in graph deep learning. Initial works in the domain of graph networks define convolution [13] and LSTM operators [58] on graphs. These methods have tried to utilize node linkage information, Fourier domain in graphs etc and shown promising performance in different tasks of graph based data. Recently, other deep leaning networks such as attention networks, auto-encoders, generative networks [11, 93, 76] etc have also been defined/extended in the domain of graphs. Our focus in this works remains exclusively in the domain of graph convolution networks. Furthermore, graph convolutions can be defined via a spectral approach [13, 21, 48, 56] or a spatial approach [30, 75, 22, 16]. Since our previous work on image-annotation in Chapter 3 uses spectral formulation, we use graph convolution networks defined in the spectral domain.

4.0.1 Our Contributions

- We propose a graph signal based deep-learning solution for the problem of multi-label classification
- Through experimentation of our architecture, we explore the effects of graph formation and the input signals on the performance.

4.1 Background

In this section we discuss the formulation from [21] which is used in this chapter and is a continuation of subsection 2.3.2. Equation 2.11 being a non-parametric filter suffers from (i) non-localized in space (ii) learning capacity is $\mathcal{O}(n)$(dimensionality of data). These are resolved by defining a polynomial filter:

$$\hat{h}_\theta(\mathbf{\Lambda}) = \sum_{k=0}^{K-1} \theta_k \mathbf{\Lambda}^k \tag{4.1}$$

where $\theta \in \mathcal{R}^k$ is a vector of polynomial coefficients. [37] shows that this polynomial parametrization makes the filter K-localized and also reduces the learning complexity of the filter to $\mathcal{O}(K)$ (support size of filter).

Since the cost of matrix multiplication on $\boldsymbol{U}\hat{h}_\theta(\mathbf{\Lambda})\boldsymbol{U}^\top$ is $\mathcal{O}(n^2)$, the polynomial parametrization of Equation 4.1 needs to replicated to $\hat{h}_\theta(\boldsymbol{L})$. [37] shows that one way of achieving this can be by formulating a recursive solution(from $\boldsymbol{L}$) for the polynomial parametrization. This is knows as Chebyshev expansion and is given by:

$$\boldsymbol{y} = \hat{h}_\theta(\boldsymbol{L})\boldsymbol{s} = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\boldsymbol{L}})\boldsymbol{s} \tag{4.2}$$

Here $\theta \in \mathcal{R}^k$ is a vector of Chebyshev polynomials, and $T_k(\tilde{(\boldsymbol{L})} \in \mathcal{R}^{n \times n})$ is the Chebyshev polynomial of order k evaluated at the scaled Laplacian $\boldsymbol{L} = 2\boldsymbol{L}/\lambda_{max} - \boldsymbol{I}$ ($\boldsymbol{I}$ is identity matrix and λ_{max} is the maximum eigenvalue). Denoting $\bar{s}_k = T_k(\tilde{\boldsymbol{L}}x \in \mathcal{R}^n$, we can use the recurrence relation to compute $\bar{s}_k = 2\tilde{\boldsymbol{L}}\boldsymbol{s}_{k-1} - \bar{s}_{k-2}$ with $\boldsymbol{s}_0 = \boldsymbol{s}$ and $\boldsymbol{s}_1 = \tilde{\boldsymbol{L}}\boldsymbol{s}$ The entire filtering operation then becomes:

$$\boldsymbol{y} = \hat{h}_\theta(\boldsymbol{L})\boldsymbol{s} = [\boldsymbol{s}_0, \ldots, \boldsymbol{s}_{K-1}]\theta \tag{4.3}$$

The entire cost of operations for this parametrization then becomes $\mathcal{O}(K|E|)$.

The final output feature map for a sample t is given by:

$$\boldsymbol{y}_{t,j} = \sum_{i=1}^{F_{in}} \hat{h}_{\theta_{i,j}}(\boldsymbol{L})\mathbf{s}_{t,i} \in \mathcal{R}^n \tag{4.4}$$

where the $s_{t,i}$are the input feature maps, $\boldsymbol{y}_{t,j}$ is the j^{th} output feature map and the $F_{in} \times F_{out}$ vectors of Chebyshev coefficients $\theta_{i,j} \in \mathcal{R}^k$ are the layers trainable parameter. We utilize Equation 4.4 to define convolution filter operation, henceforth in this chapter.

4.2 Proposed Approach

In this work, as mentioned before, we focus on finding an alternative approach to label modelling i.e. to model the label co-occurrence and dependency patterns. Recent works have approached this by modelling labels as a sequence of words [98, 62] or by using graph based models [66] for propagating labels in a controlled way. Motivated by the recent developments in the graph deep learning, we focus on finding an alternative to [66]. Our proposed approach is depicted in Figure 4.1. Similar to the recent annotation models our models also utilizes the three modalities. Now, we describe them in detail:

4.2.1 Label Modelling

Our label model is graph based i.e. each node on the graph represents a label and the edges capture the interrelations between them. Instead of explicitly expanding nodes as in [66], we focus on intrinsically capturing the flow of signals between different nodes of the graph by utilizing the spectral graph convolution formulation of [21]. For our purposes, we require two inputs (i) a knowledge graph(for defining $\mathbf{L}$) which remains static for all the samples enabling a single eigen-decomposition of the laplacian matrix and hence a constant graph Fourier basis. (ii) a graph signal which changes for each input signal. We now describe these inputs in detail:

4.2.1.1 Knowledge Graph

In most graph based applications, the knowledge graph (or adjacency matrix) is pre-defined, which, however, is not provided in any standard multi-label image recognition datasets. Capturing patterns of label dependence and co-occurrence in labels could be achieved by quantifying pairwise relation between them. Since edges on the graph captures pairwise relationship, we use it to capture the label relations.

We use different sources of information to construct our knowledge graph. To define a graph we need to define the nodes and edges(with weights). Our main idea to define edge weights is to count co-occurrence among labels. Using the train split of the dataset we count co-occurrences between each set of pairwise labels, and then discard all edges below a certain threshold. In particular we use 2 strategies to weigh edges and define nodes in the knowledge graph:

- MSCOCO Graph: The node set in this graph consist of all the labels in the MSCOCO dataset ie. 80 nodes. We use the trains set of MSCOCO dataset to define the edge weights. We set the pairwise occurrence threshold to 400, below which all edges are discarded.
- VGML enhanced graphs: We use Visual Genome[46] as an external source for our knowledge graph. It consists of $100,000$ natural images which is annotated as objects, attributes and the relationships between them. We follow [66] to obtain 316 labels which forms the nodes of the graph. These 316 labels consists of MSCOCO labels, 200 most frequently occurring objects 100 most frequently occurring attributes in Visual Genome (MSCOCO labels overlap with objects from Visual Genome). We count the object-object and object-attribute pairwise occurrences and prune edges having fewer than 200 instances. This knowledge graph is an external source of information hence the model with this graph remains as a local approach in image annotations.

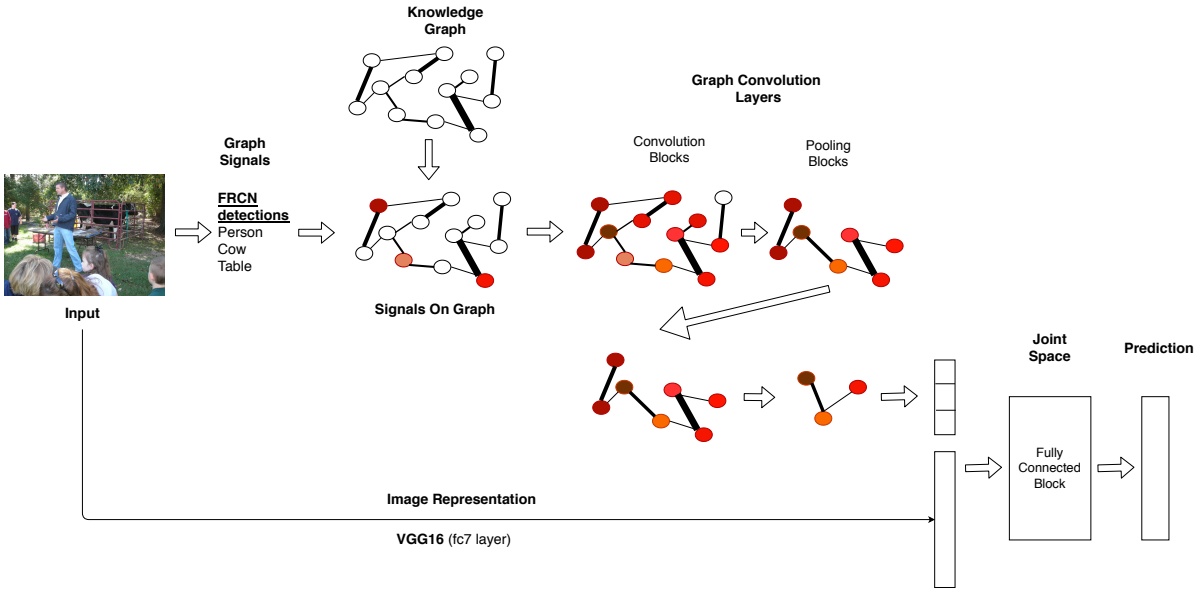


Figure 4.1: Proposed approach: Given an input image, its representation is computed by passing it through VGG16(fc7 layer) network with pretrained weights. On other hand Faster-RCNN detections are passed as input signals to graph which undergoes through 2 stages of graph convolution and pooling layers(as defined in [21]. Later both the above representations are concatenated and passed through a fully connected neural net before final prediction. Note that colors denote activated(non-zero) signals and the width of the edges denote different edge-weights. Notice that one of node is is inactive after 1^{st} convolution layer. This depicts that the diffusion of the signals is limited in the framework due to the choice of the parameter K(=2 here) in Equation 4.3 (Best seen in color)

We ensure that our graph is connected, so that the signals may diffuse to all nodes/labels. This ensures that any two nodes which do not have an edge between them may still influence one-another by the virtue of a path between them. In our initial experiments, we enhanced the edge weight by using cosine similarity between vector representations of labels, but the resulting performance was poorer.

4.2.1.2 Graph Signals

During training and testing, we need to provide a signal($x \in \mathbb{R}^n$) to each graph-node. We determine the value on these signals based on likelihood of the concept being present in the image as determined by an object detector or a classifier. For our experiments, we use Faster R-CNN(FRCN) [77] for each of the 80 MSCOCO categories. Instead of passing FRCN likelihood for all 80 categories, we binarize these values above a certain threshold. Unless otherwise specified we set the this threshold at 0.5.

4.2.2 Image Representation

Following the trends of recent works of [62, 98] we use VGG($fc7$ layer) to extract image representation. This choice of network also help us in a direct comparisons of the models. We chose not to fine-tune this part of the network as we focus on capturing the label dependencies for the graph convolution networks.

4.2.3 Joint Space

Instead of learning a joint space to project the representations from the above stage, we chose to learn a network to combine the signal representation and the image representation. Following the work of [66] we concatenate the representation of both the previous modalities and pass it through a fully-connected network. Introducing a fully-connected block allows the whole model to be fully-differentiable. For the prediction we chose top-r labels with highest scores. The value of r is set as the mean labels per image in the dataset(rounded off to the closest integer). In case of MSCOCO we set r as 3.

4.2.4 Network Architecture

Here, we provide detail about the Graph Convolutional Layers and Fully Connected Block as shown in Figure 4.1. We stack 2 layers of graph-convolution, each followed by a pooling layer (as defined in [21]) and an activation function (leaving the last layer). Similarly for Fully connected block we chose 1 or 2 hidden-layers before the final prediction. Each such hidden layer in followed by an activation function.

4.2.5 Training and Network Parameters

Although our model is end-to-end trainable we train it in a greedy manner (fixing the weights for the detector network- FRCN and for image representations-VGG). We set an initial learning rate of $1e-4$ while using an Adam optimizer, weight regularization of $5e-4$ for all parameters of graph convolution layers, batch-size of 16, and trained all our models till convergence (which turned out around 20 epochs for most experiments). We set K=2 (Equation 4.4), number of filters(Equation 4.4) as 16, 32 respectively in graph convolution layers and the pooling scale 1 (i.e after each graph convolution block all the nodes

are retained). Relu was utilized as the activation function in Graph Convolution layers and in fully connected block. We use binary cross entropy loss to train our network.

4.3 Dataset

4.3.1 MSCOCO

2Microsoft COCO (MSCOCO) dataset [60] is an image-recognition, segmentation, and captioning dataset. It contains $123,287$ images pertaining to 11 super-categories which are further hierarchically divided into 80 objects categories. The images are predivided into two sets: training set and testing set. $82,783$ images are utilized as training set images while $40,504$ images are employed as testing set images(Furthermore, $40,775$ images are hidden are used for the online leader-board for detection and captioning tasks. To obtain labels we utilize the object annotations. On average each image is labeled with 2.9 tags and hence this forms a suitable dataset form multi-label image annotation.

The labelling of the MS-COCO dataset is done in three stages: (i) Super-Category labelling: In this stage the annotators were asked to verify if a specific super-category was present in the image. (ii) Instance spotting: Here, annotators were asked to manually mark(a cross symbol) and annotate the object category for each instance(found in the previous stage). (iii) Instance Segmentation: Here, annotators were asked to segment an object instance as detected in the previous stage. The labelling process was also supported by a verification stage where annotators discarded poorly segmented images and the annotation task was repeated for these images. This ensured good quality labelling i.e. the dataset was void of incomplete labelling(un-labelled objects) and wrong labels. An interesting property of the MS-COCO dataset is that most images in this dataset contain multiple objects, and these objects usually have strong co-occurrence dependencies. For example, baseball glove and sport ball have high co-occurrence probability, while zebra and cat never appear together.

The 11 super-categories are: Animal, vehicle, outdoor-objects, sports, kitchenware, food, furniture, appliances, electronics, indoor-objects and person & accessory. The 80 object categories are: person, chair, banana, handbag, skis, skateboard, bicycle. couch, microwave, umbrella, teddy-bear, hot-dog, snowboard, car, sink, toilet, apple, scissors, tie, potted-plant, motorcycle, tv, oven , book, keyboard, truck, fire-hydrant, stop-sign, sports-ball, bird, cow, toaster, boat, backpack, traffic-light, bowl, refrigerator, surfboard, broccoli, donuts, cat, airplane, bus, pizza, sandwich, suitcase, vase, dog, train, cellphone, wine glass, cup,tennis-racket, basket-ball, fork, carrot, cake, basketball-glove, kite, knife, mouse, horse, sheep, bed, dinning table, remote,zebra, hairdryer, spoon, Frisbee, orange, parking-meter, giraffe, bottle, elephant, laptop, clock, bear, toothbrush and bench.

4.4 Experiments

We conduct a few experiments to test various network architectures and parameters in the graph convolution block. Through these experimentation we try to check for the robustness of the graph convolution formulation. Apart from this we also compare our best results with other comparable methods. We now provide the evaluation metrics and the baseline used in this work:

4.4.1 Evaluation Metrics

We report Precision(P), Recall(R), F1 measure(F1) and mean Average Precision(mAP) for all our experiments. For a detailed explanation about the metrics, refer Section 3.3.1 and for detailed discussions about metric behaviour/comparisons refer Section 3.3.3.

4.4.2 Baselines

Similar to [66] we define a baseline devoid of label modeling. Specifically, we concatenate VGG16 representation with FRCN detections and pass it through the 2 layers(1024 and 512 neurons in the hidden layers respectively) of fully connected block before final predictions. We refer to this baseline as **VGG+Det** in further sections.

4.5 Results and Discussions

4.5.1 Quantitative Evaluation

4.5.1.1 Knowledge Graphs and Input Signals

Table 4.1: Ablation Study: Performance variation due to different knowledge graph and graph signal in our model. In row-5, the graph input signal is **not** binarized after thresholding.

Method	FRCN threshold	Knowledge Graph	P	R	F1	mAP
VGG+Det(baseline)	-	-	23.71	42.33	30.36	73.93
Ours	0.9(binary)	VGML-enhanced	36.35	60.7	45.47	74.0
	0.5(binary)	VGML-enhanced	37.23	62.57	46.68	74.2
	0.9(binary)	MSCOCO-graph	40.2	68.49	50.66	74.5
	0.5(non-binary)	MSCOCO-graph	41.15	67.61	51.16	75.7
Ours(best)	0.5(binary)	MSCOCO-graph	43.59	72.41	54.41	76.5

Table 4.1 shows results of ablation study due to different knowledge graphs and graph signals. We make the following observations:

- We observe that MSCOCO graph works better than the VGML-enhanced graph consistently over all the metrics(Row-2 vs Row-4 and Row3 vs Row 6). This verifies our ideology that a combination of global and local approaches is better than purely local or global approach. It may be possible to design a generic-graph which is devoid of its dependency on the used data but doing so may require manual enhancements.
- For graph input signals, higher threshold in FasterRCNN detections degrades the performance for both VGML-enhanced and MSCOCO graphs(Row-2 vs Row-3 vs Row 4 vs Row-6). With higher thresholds fewer nodes have non-zero values(in graph signals) which may lead to incomplete diffusion.
- We also observe that the performance from non-binarized input signals (for graphs) is poorer then binarized signal(Row-5 vs Row-6).

We hypothesize from the above observations that graph based deep learning approaches are susceptible to underlying knowledge graphs (or adjacency matrices).

4.5.1.2 Comparison with other methods

Table 4.2: Comparison of popular methods on different evaluation metrics for MSCOCO dataset

Method	P	R	F1	mAP
CNN-RNN [98]	69.2	66.4	67.8	61.2
S-CNN-RNN [62]	77.41	73.05	75.16	-
GSNN-VGG [66]	-	-	-	77.57
VGG+Det(baseline)	23.71	42.33	30.36	73.93
Ours	43.59	72.41	54.41	76.5

Table 4.2 provide a comparison of different popular label-modelling methods from the literature. Row $1, 2$ consists of methods that uses RNNs to model labels, whereas Row 3 uses graph LSTM networks to model labels. Our observation in Section 3.3.3 still remains valid: "the obvious understanding one can make here is that there is non-agreement between F1 and mAP measures". While comparing with [98, 62],we observe that our method is far behind in precision and F1 but is far better in mAP and performs equivalent in recall. We also observe that the our method outperforms the baseline on all the metrics. While comparing with [66] we observed similar performance in the mAP scores[1].

One of the potential reasons as to why our precision, recall and F1-scores are effected might be due to a fixed number of prediction per sample(3 in the case of MSCOCO) which increases the tags belonging false positives and false negatives. Models utilizing RNN and graph LSTMS can predict a variable number of samples and hence have fewer tags missing.

[1]At the time this work was done, authors of [66] did not release the code and were unresponsive on mail

4.5.2 Qualitative Results

Figure 4.2 shows the qualitative results on a 4 randomly selected samples from the MSCOCO dataset. The model misses on some of the obvious tags such as *scissors* and *bottle* in bottom-right and *kite* in bottom-left. This may also be possible as our model is fixed to predict 3 tags per sample. Detecting a few tags might be tricky(as the tags are barely visible) as is the case with *car* in top-right.

Figure 4.2: Qualitative results for label prediction on MSCOCO dataset. 4 randomly selected samples and their predictions and ground truth labels(color coded). Green: Labels present in ground-truth and our model's predictions (true positives), Blue: Incorrect predictions by the model (false positives). Red: Missed Predictions (false negatives) (Best seen in color)

Chapter *5*

View Invariant Trajectory Mapping for Autonomous Driving

We take videos or image sequences as the input to perform the recognition task. The temporal information present inherently in such ordered image sequences leverages neighborhood information between frames, establishing relationships among objects in the scene. This also helps to mitigate the problem of perceptual aliasing to some extent, a problem that often plagues image-based recognition/relocalization approaches.

We propose a novel stacked deep network ensemble architecture that combines state-of-the-art CNN, Bidirectional LSTM and Siamese style distance function learning for the task of view-invariant intersection recognition in videos. While the CNN component of the ensemble primarily deals with image-level feature representation, the bidirectional LSTM is the key recurrent network component that enables learning of temporal evolution of visual features in videos followed by the Siamese network-based distance metric learning for comparing two input video streams. Our proposed network is conceptualized in Figure 5.1.

5.0.1 Contributions

In this published paper we contributed in the following ways:

1. Firstly, we propose a novel and pertinent problem of recognizing intersections when approached from highly disparate viewpoints

2. We propose an original deep network ensemble. The proposed architecture can handle videos of varying length and compare videos capturing reverse trajectories. This kind of Siamese network with recurrent LSTM component has been largely unexplored in the video domain. Furthermore, we use the hidden state of LSTM cells, instead of the traditionally used output state, for video representation, which has largely been unexplored too.

3. We showcase the efficacy of the proposed architecture through competitive results on challenging sequences while showing decent improvements from the image based baselines. We collect and annotate synthetic yet highly realistic data from the GTA [2] gaming platform and actual real world data of Mapillary [4] for this purpose.

To the best of our knowledge, this is the first attempt in this direction using the ensemble of state-of-the-art CNN, bidirectional LSTM and Siamese network architecture for recognizing intersection in input video pairs.

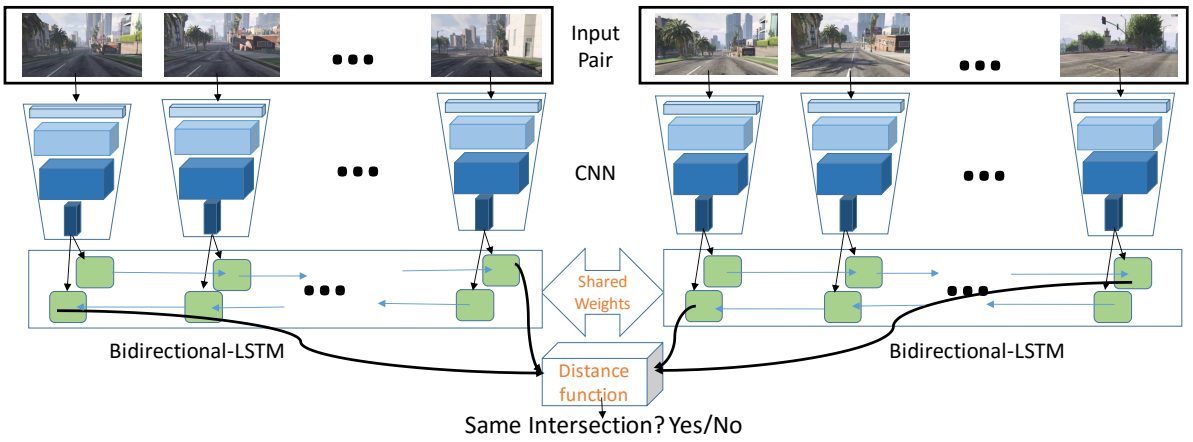


Figure 5.1: Proposed model is shown with video pairs as input and binary-classification as the output. Features from pretrained CNN network are fed into bidirectional LSTM with shared-weights as shown with hidden units in green (unfolded in time).

5.1 Proposed Approach

In order to achieve view-invariant intersection recognition from video sequences, our overall approach follows the recent successes of deep network models in learning useful representations of visual data [9]. Our overall strategy comprises of three specific sub-objectives: (i) capture representations of individual video frames that can help identify the intersection in a view-invariant manner; (ii) leverage the temporal correlations between video frames to better capture the unique identity of an intersection across different views; and (iii) use an appropriate distance metric to compare the thus learned spatiotemporal representations of intersection video sequences. We use a stacked deep network ensemble architecture to realize the above strategy.

Our stacked ensemble consists of a state-of-the-art Convolutional Neural Network (CNN) to learn video frame representations, a Bidirectional Long Short-Term Memory (LSTM) Network (a variant of Recurrent Neural Networks, that addresses the vanishing gradient problem) that captures temporal correlations across video frames, and a Siamese Network that learns a suitable distance metric that can uniquely identify an intersection from these video sequences across views. Figure 5.1 summarizes the proposed ensemble strategy.

5.1.1 Convolutional Neural Networks (CNNs)

CNN based models have shown tremendous success in a variety of tasks like [50], [7], etc. and it has been shown that representations learned by the CNN can be used as off-the-shelf features for other tasks [83], [89].

From a plethora of CNN pretrained models, we choose the model most relevant to our tasks which provide invariance in lighting and pose. In particular, we employ AmosNet [17] as our CNN architecture. AmosNet is a variation of CaffeNet [50] trained for Visual Places Recognition under varying weather conditions. We use pooled outputs from last convolution layer of AmosNet as the CNN features.

5.1.2 Bidirectional LSTM

Long Short-Term Memory Network (LSTM) [39], is a variant of Recurrent Neural Network (RNN) with the ability to capture long-term dependencies [47]. It has a hidden representation which is updated at each time step, and also consists of a memory state which holds the relevant information at each time step. The information entering and exiting out of cell states is controlled by gating mechanisms (sigmoid, and $\tanh$ functions).

Bidirectional LSTMs [82] incorporate future and past context by running the inverse of the input through a separate LSTM. The proposed model employed Bidirectional LSTMs capture the temporal correlations between video frames. This enables our model to identify the intersection uniquely whether a vehicle approaches it in the forward or opposite direction. Instead of using averaged/fused output of unfolded LSTM units or output from last unit we use the hidden-representation from the last unfolded time step as a feature vector for the videos as shown in Figure 5.1.

5.1.3 Siamese Network

The third component of our deep network ensemble is a Siamese network, which is used to learn data-driven distance metrics. A Siamese Network [103] consists of two independent networks with same architecture and shared weights. Both these networks are merged to learn a distance metric, d, between the two inputs provided to the respective networks.

During training, Siamese networks work on triplets ($\mathbf{x}_1$, $\mathbf{x}_2$, y) where y is the ground truth similarity between $\mathbf{x}_1$ and $\mathbf{x}_2$, i.e., $y = 1$ if the videos denote the same intersection, else $y = 0$. The networks' weights are then updated by minimizing the loss function described below.

Contrastive Loss: The total loss function over a dataset $X = \{(\mathbf{x}_1^i, \mathbf{x}_2^i, y^i), \forall i = 1, \cdots, n\}$ is given by:

$$L_{\boldsymbol{W}}(X) = \sum_{i=1}^{n} L_{\boldsymbol{W}}^i((\phi(\mathbf{x}_1^i), \phi(\mathbf{x}_2^i)), y^i) \tag{5.1}$$

where $\mathbf{W}$ are the weights/parameters of the network, $\phi(\mathbf{x})$ denotes the output of the last layer of the shared network architecture for each individual input $\mathbf{x}$, and $L^i_{\mathbf{W}}$ is

$$L^i_{\mathbf{W}} = y^i L_{pos}(\phi(\mathbf{x}^i_1), \phi(\mathbf{x}^i_2)) + (1 - y^i) L_{neg}(\phi(\mathbf{x}^i_1), \phi(\mathbf{x}^i_2)) \tag{5.2}$$

where

$$L_{pos}(\phi(\mathbf{x}^i_1), \phi(\mathbf{x}^i_2)) = d(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2))^2 \tag{5.3}$$

$$L_{neg}(\phi(\mathbf{x}^i_1), \phi(\mathbf{x}^i_2)) = \max(1 - d(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)), 0)^2 \tag{5.4}$$

At test time, the learned distance function is used to predict the similarity in the videos using the expression in Eq. 5.5.

$$\hat{y} = \begin{cases} 0 : d(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)) > \theta \\ 1 : d(\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)) \leq \theta \end{cases} \tag{5.5}$$

We set the value of θ to be 0.5 in our experiments. Since the number of possible negative pairs can be very high as compared to positive pairs, we scale the loss function for each positive by a constant factor (> 1), given by the ratio of negative pairs to positive pairs in the dataset.

5.1.4 Training and Network Parameters

Although our model can be trained in an end to end manner we train it in a greedy way due to negligible effect on performance as compared to end-to-end training. We use pretrained weights from previous state-of-the-art CNN models which provide a meaningful weight-initialization to our model and reduces training time. We use the contrastive loss described in Eq. 5.1 to train the Bidirectional LSTM, as shown in Figure 5.1. The proposed model is trained using the ADAM optimizer which is a variant of Stochastic Gradient Descent (SGD).

In our initial experiments, we varied the number of Bidirectional LSTM-layers (upto 3), but the performance gain was negligible. Similarly, we experimented with different values for hidden-unit dimensions in the LSTM cell. Empirical results found 250 to be the best performing after which the performance saturates. Thus, we fix the number of Bidirectional layer to one and hidden layer dimension to be 250 in all further experiments.

5.2 Experiments & Results

In this section, we first define the baselines used for comparison with our proposed method. Next, we proceed to provide description about synthetically generated and real datasets for various experiments. Subsequently we describe the various experimental scenarios which is followed by the explanation of the quantitative and qualitative results.

5.2.1 Baselines / Evaluation Settings

In this subsection, we discuss two different baselines for comparisons with our proposed method.

Siamese-CNN: We define a deep learning approach void of extrinsic temporal modeling using Siamese Network on CNN. Prior work on Siamese-CNN have included application for face-verification [91], one shot recognition [49] etc. First, we train a Siamese-CNN using positive and negative pairs of images depicting same and different intersections respectively. Then, given two videos ($\mathbf{x}_1$, $\mathbf{x}_2$) comprising of n and m frames respectively, we use learned Siamese-CNN for prediction (using Eq. 5.5) for all possible frame comparisons (nm scores) and combine these predictions into a single score by computing their sum. We then learn a threshold maximizing the accuracy. During testing we then use this threshold in Eq. 5.5 to predict similarity in videos. For fair comparison the Siamese-CNN is initialized with same pretrained weights.

X-View: For the purpose of comparison with recent works on drastic-viewpoint changes we focus on [31]. Since [31] uses multiple data-modalities, for a fairer comparison, we limit the data-modalities to those strictly necessary for the model, using robot odometry (GPS coordinates and heading changes) and semantic-segmentation on RGB images. We ignore the backend needed in [31] as we do not focus on localization task, instead replacing it with a match pruning procedure. Each node in the query graph votes for k frames in the memory graph, based on matches with component nodes of these frames. Windows of frames in the memory sequence covering a threshold of votes from the distinct nodes/frames from the query graph, are taken as candidate matches corresponding to the query sequence.

We found this method to be prone to $semantic$ perceptual aliasing, as semantic blobs of the same class are non-discriminative as their neighboring nodes' relations are their only descriptors. Thus, a direct comparison of [31] in the same context as ours (one shot recognition, instead of relocalization within a previous traversal sequence) wouldn't be meaningful.

Figure 5.2: Data Visualization: Random snapshots from GTA environment [2] (Row 1) and Mapillary [4] (Row 2). Images show different weather and day/night conditions as well as various outdoor scenes for urban scenarios. The game-environment snapshots look visibly similar to that of real world images and have significantly higher variations in weather and lighting. (Best seen in color)

Table 5.1: Number of video pairs in training, testing and the validation set for the different experimental scenarios

Dataset →	GTA		
Lighting Setting →	day and night		day
Trajectory Relations →	All combinations	Overlap in view and trajectory	All combinations
Train	1509	640	864
	1440	1440	1166
Val	459	168	300
	612	212	653
Test	106	62	72
	104	104	82

Dataset →	Mapillary	
Lighting Setting →	day	
Trajectory Relations →	All combinations	Overlap in view and trajectory
Train	3080	6409
	3328	6976
Val	421	350
	428	318
Test	400	425
	400	420

5.2.2 Dataset Description

Dataset Collection: Each video/sequence consist of a series of frames collected around junction points. Each junction can be traversed in multiple pathways. We refer to the set of all such possible traversals as *trajectories*.

5.2.2.1 GTA

Here, we collected videos in two different scenarios. Firstly, we choose a set of intersections in the game-environment and then sample them in a *dense* manner, i.e., each possible trajectory in the junction is traversed. Secondly, we traversed a car from one point to another via means of an AI-car driving mod [3]. From such traversals, we break the captured video into small chunks involving intersections and non-intersections. We discard video chunks without intersections.

In the first scenario, we selected 27 unique junctions where 9 of them had arbitrary lighting conditions and the rest were in normal daylight condition. In the second scenario, we captured 12 traversals in two arbitrary different lighting variations.

5.2.2.2 Mapillary

We download images from [4], which is a community-led service with more than 200 million street-level images. These images are captured in various modes including walking, riding (either a bike or car), panorama and photo-spheres. We use mapillary's API to download images and construct *trajectories*. We first query for all the images in a bounding box defined by the longitudes and latitudes. For every image in the bounding box, the API provides a tuple consisting of image-key, latitude, longitude, orientation and the (video-)sequence it belongs to along with other metadata information. These images can be then downloaded using the image-key. Using two orthogonally overlapping sequences we get a location in the map which is used to identify the intersection. Subsequently, this is used to download all sequences (a set of continuous images) passing through that intersection. We mined around $300,000$ images which consisted of around 1700 usable sequences from around 500 junctions.

Figure 5.2 shows random snapshots captured from GTA environment (Row 1) and Mapillary (Row 2). One can infer from the figure that the snapshots from GTA environment are visually realistic as well as similar to real-world data from Mapillary and contain the key challenges associated with real environments (as listed in Section 1.3). However, the GTA environment offers more complexity in terms of weather and lighting variations as compared to Mapillary.

Dataset Annotation: From the extracted dataset, we first annotate positive and negative video pairs. Positive (videos involving the same intersection) pairs are mined from the above datasets exhaustively, i.e., we select all possible positive pairs. Negative pairs generated from the datasets can be very high as any two trajectories from different junction are treated as a negative pair. Hence we limit the number of negative pairs by randomly fixing a subset. We keep the test, train and validation sets mutually exclusive by sampling them from different junction and/or non-overlapping traversals.

Table 5.2: Accuracy, Precision, Recall and F1 of our method on different datasets for the different experimental scenarios

Metrics ↓	Dataset →	Mapillary	
	Lighting Setting →	day	
	Trajectory Relations →	All combinations	Overlap in view and trajectory
Accuracy	Our Method	**72.1**	**81.0**
	Siamese CNN	63.0	71.6
F1 Score	Our Method	**61.9**	**82.4**
	Siamese CNN	428	318

Metrics ↓	Dataset →	GTA		
	Lighting Setting →	day and night		day
	Trajectory Relations →	All combinations	Overlap in view and trajectory	All combinations
Accuracy	Our Method	**70.95**	**78.2**	**76.72**
	Siamese CNN	65.3	65.3	68.0
F1 Score	Our Method	**62.3**	**60.0**	**72.7**
	Siamese CNN	47.9	52.8	51.1

5.2.3 Experimental Setup

For a pair of videos involving the same junction there exists a trajectory-relation between them, which can be categorized in terms of overlap in view and/or trajectory. Overlap in trajectory refers to the proximal relation in the two trajectories while overlap in view refers to trajectories, viewing the larger part of the same area. For example trajectories moving in opposite direction can be proximal in distance but can have minimal view overlap.

For the purpose of our experiments we categorize the trajectory-relations into two primary setups: *Overlap in view and trajectory* and *All combinations*. The former refers to parallel and overlapping trajectory-relations while the latter refers to all the possible trajectory-relations. Similarly, we categorize the lighting setting into two categories: *day* vs *day and night*. The names are self-explanatory.

We test the robustness and generalizing capability of our model under various trajectory-relation and lighting (day/night) settings. The details of the experiment-wise distribution of samples can be found in Table 5.1.

Evaluation Metric: We report percentage accuracy and F1 as the metric in our experiments. Accuracy is defined as the ratio of the number of correctly classified samples (both positive and negative pairs combined) to the total predictions made. F1 is defined as the harmonic mean of precision and recall. We note that while accuracy focuses on both positive and negative predictions, F1 is focused toward positive predictions (as precision and recall focuses on true-positives).

5.2.4 Quantitative Results

Table 5.2 shows performance on various metrics on GTA and Mapillary dataset under varying scenarios using our proposed model and baseline Siamese-CNN. We show different comparisons based on lighting setting, trajectory relation and baseline approach.

GTA: We observe that our proposed method performs better than the Siamese-CNN baseline on both Accuracy and F1. Additionally, among different models in the proposed approach (for different lighting conditions) we observe that the model improves on accuracy (by 5.76%) and F1 (by 10.4%) in *day* scenario as compared to *day and night* scenario. Similar trend of values is also observed in the Siamese-CNN baseline. This trend of results can be attributed to the reduced complexity in lighting variations.

Interestingly, under the *day and night* condition, the model performs better at accuracy (7.25% increase) when video-pairs have an *Overlap in view and trajectory* as compared to *All combinations*. However we observed an inverse trend for F1 measure. This anomalous behavior can be explained based on the performance on negative samples in this scenario. In this case we found that the true negative rates performance for *Overlap in view and trajectory* is 83.0% as compared to *All combinations* at 52.8% (30% increase). Nevertheless, the performance of proposed model is still significantly higher than baseline method.

Mapillary: Similar to GTA, we observe that our proposed method performs better than the Siamese-CNN baseline on both accuracy and F1. Among models in our proposed method, the performance is better

Table 5.3: X-View results on Mapillary: F1-scores at different experimental threshold (query sequence length: 15)

Localization Threshold	Inlier Frames %	F1
20m	70	51.5
40m	70	50.9
40m	80	47.7
60m	70	43.7
60m	80	46.7

on all metrics (accuracy: 8.9%increase, F1:20.5%increase) when the trajectory-relations are limited to *Overlap in view and trajectory* as compared to *All combinations*. Similar change in values can be observed in the Siamese-CNN baseline.

GTA vs Mapillary (for proposed method): In the common scenario of *day* and *All combinations*, we notice that the performance on GTA is better than Mapillary for accuracy (4.62% increase) and F1 (10.8% increase). This can be explained due to complexity of data annotation in real-world where visually dis-similar videos can be annotated as same based on their GPS coordinates.

Table 5.3 shows experimental results of X-view on Mapillary in relocalisation scenarios. We used 12 relevant and reliably segmented semantic categories. Localisation Threshold(maximum distance from Ground Truth for a match) and Inlier Frames%(% of dictinct valid frames voting for a window of frames in the memory sequence) were varied to obtain the best F1 scores. We observe that F1 scores obtained are always lower than F1 scores for Mapillary data in Table 5.2. The method had high recall but low precision across multiple thresholds. This can partly be attributed to the fact that [31] requires 15 or more overlapping frames for reliable localization whereas in case of orthogonal trajectories in Mapillary only 5-7 frames have overlapping structures.

5.2.5 Qualitative Results

Figure 5.3 shows 3 positive samples from Mapillary that are successfully predicted by the network. Each sample depict a unique trajectory-relation and view-overlap as shown by the *intersection-map* (bottom-most row). We color encode (green, cyan, red, brown) the visually perceptible common structures (using ellipses) in the trajectories. We follow the same encoding while plotting the respective *intersection-map* in the last row. In video pair 1 and video pair 3, we observe that the overlapping part of trajectories are in the opposite direction but our network managed to exploit the common structures seen from different views. In video pair 2, though the partially overlapping trajectories are in the same direction, we can observe that *shadows* pose a challenge that is successfully handled by the network. This indicate the ability of proposed network to generalize on varying lighting conditions. Moreover, we also observe occlusion in a few trajectories (video pair 3 trajectory 2) due to vehicles.

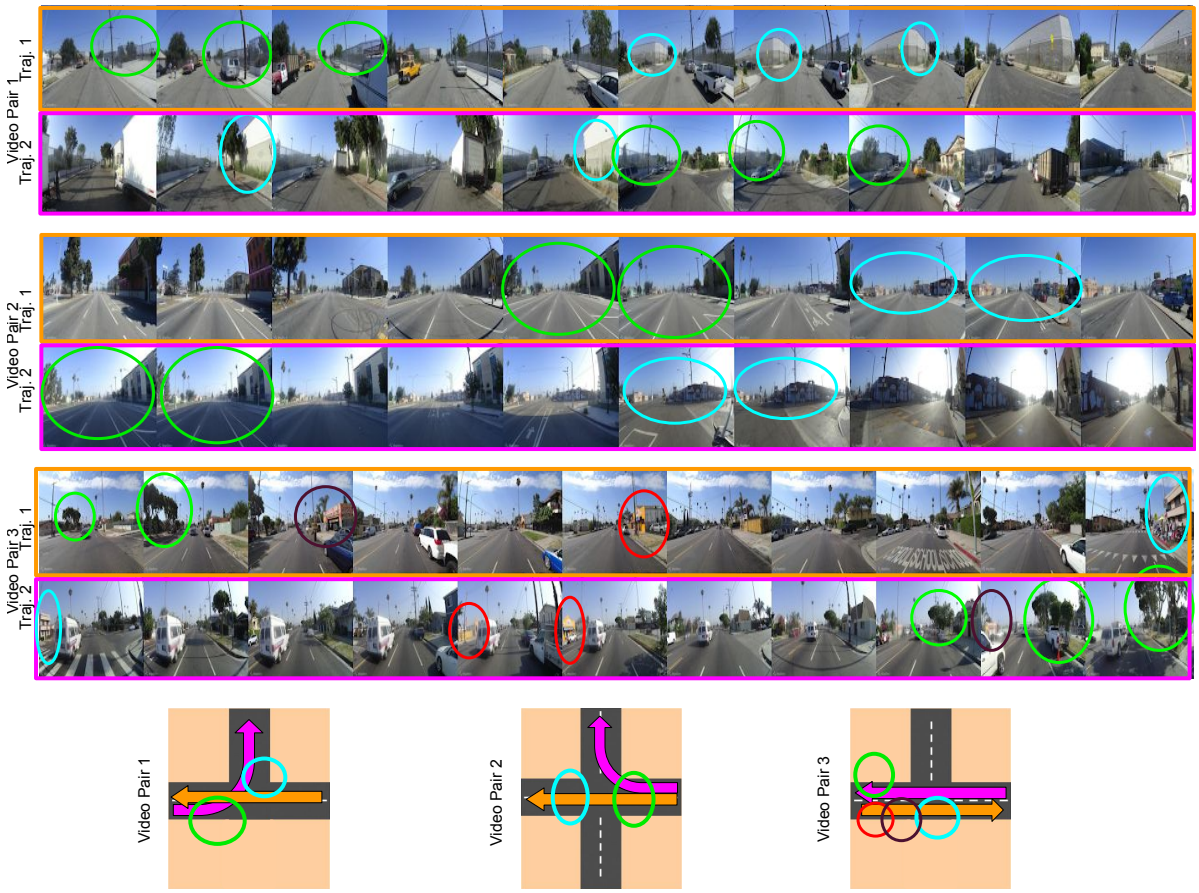


Figure 5.3: Three correctly classified samples from Mapillary. Each sample depicts unique trajectory-relation and view-overlap between trajectories as shown in the *intersection-map* (bottom-most row) which represents the *top-view* of the intersection. We color encode the visually perceptible common structures (cyan, green, red, brown) in the Video pairs. Video pair 2 and Video pair 3 show examples where trajectory direction and view overlap. Video pair 1 depict cases when the overlap in trajectories and/or the view is minimal. (Best seen in color)

Figure 5.4 depicts 2 positive samples from GTA that are successfully predicted by the network. For example: video-pair 1, capture the different view of the same structure in rainy weather. Similarly, in video-pair 2 the model is able to predict similar videos in presence of view and lighting variations. Due to space constraints we only show 6 aligned contiguous frames in these video-pairs, where we manually crop the relevant contiguous frames for illustration purpose.

Figure 5.5 shows some incorrect predictions made by the network. Video pair 1 consist of trajectories from different intersections (negative sample) but is predicted as the same intersection. We believe this may be due to lack of unique structures in the scene as most of it has trees. Additionally, the variance in lighting conditions (as the network was trained on *day* conditions in Mapillary) can also pose a challenge if not trained on other conditions. In contrast, video pair 2 are trajectories from the same-intersection but are predicted as different intersection. In this video pair, we observe a complete lack of visual features in

Figure 5.4: Two correctly classified samples from GTA. Video pair 2 show examples where trajectory direction and view overlap. Video pair 1 depict case when the overlap in trajectories and/or the view is minimal. (Best seen in color)

Figure 5.5: Failure Cases: Three wrong predictions by the network. Video pair 1 consist of trajectories from different intersections but is predicted as the same intersection. In contrast, video pair 2, 3 consist of trajectories from the same-intersection but are predicted as different intersection

the overlapping part of trajectories. Overlapping structures are completely occluded in the first trajectory due to the presence of *truck*. Using purely visual content for all these failure cases, even humans might easily fail without extra context (ground truth labels were generated using GPS coordinates). Similar to video pair 2, video pair 3 trajectories belong to the same intersection but are predicted as different. Biased on the grounds of pure visual content, the video-pair does not contain a common region of interest and hence can be argued that the prediction from the network is correct.

To further showcase some negative results of our model we present some more qualitative results in Figures 5.6, 5.7. Each of the video depicted here consists of 14 frames which is in contrast to Figures 5.3, 5.5 which consisted of 10-11 frames. We show more frames in these figures to reinforce the complex data, temporal and visual complexity of the problem at hand.

Figure 5.6 shows examples of false positives, that is, each video pair consist of video from different intersections but is predicted as to be same. We observe that the visual similarity makes it harder for the network to classify the trajectories as different samples. Since our method is based just on a visual method, we believe that such errors are justified/acceptable.

Figure 5.6: Failure Cases(False Positives): Two wrong predictions by the network. Video pair 1 and 2 consists of trajectories from different intersections but is predicted as the same intersection. Both the trajectories in Video-Pair 1 capture the side-view of the road traversal which adds to complexity of intersection recognition. Video-Pair 2 is in fact a true failure case and we believe sch errors are acceptable due to the sufficient visual similarity in the videos.

We further showcase 2 false negatives that is the network predicted the pair of trajectories as different while they belonged to the same intersection as shown in Figure 5.7. We hypothesize that the network failed to find the static parts (such as common buildings view points) in these trajectories and hence failed to identify them belonging to the same intersection.

Figure 5.7: Failure Cases(False Negatives): Two wrong predictions by the network. These Video pairs belonged to the same intersection but were predicted as different by the network.

Chapter *6*

Conclusion

Scene understanding entails enabling machines to understand scenes, similar to humans' understanding of it. Thus, scene understanding engulfs a wide array of problems involving visual data. Recent outburst of visual data has facilitated fast paced development of this field. In this thesis we restrict ourselves to a couple of problems namely image annotation and intersection recognition from the domain of scene understanding.

In Chapter 3, we presented a solution for automatic multi-label image annotation. Our method exploits the empirical observation that similar images have similar neighborhood-types in terms of their label distribution. We have introduced the notion of neighborhood-type and proposed to learn local MKL models per cluster/neighborhood-type with closed form learning solution. The MKL formulation exploits the multi-scale diffusion where we also proposed a novel diffusion scale normalization to be able to combine diffusion at different local graphs. Finally, we have shown promising results on publicly available dataset. Capturing higher order relations with the help of hyper-graphs and development of deep hyper-graph networks is one of the potential directions for future work.

In Chapter 4, we presented an alternate approach for label modelling using the newly rising field of graph deep networks. Our model utilizes the graph convolution formulation by Defferrard et. al.[21]. We explore weather label modelling from such graph networks formulations are comparable with traditional deep learning approaches. Further, we focused on testing the robustness of the model to different (graph)input signals and different knowledge graph formulations. Although our work here is on image annotation involving two modalities of data: images and labels, we believe that hybrid- deep learning models(i.e. models with graph networks and regular grid networks) can be used to model multiple modalities of data and combine them efficiently.

In Chapter 5, we proposed a stacked deep network ensemble architecture that combines state-of-the-art CNN, bidirectional LSTM and Siamese style distance function learning for the task of view-invariant intersection recognition in videos. The proposed architecture enables recognizing the same intersection across two videos of variable length having large view variations, inverted trajectory, lightning and weather variations. We have collected annotated data (more than 2000 videos) from GTA [2] and Mapillary [4] and have computed results on this data with varying parameter choices and have reported competitive results.

Related Publications

1. **MKL based Local Label Diffusion for Automatic Image Annotation** [51]
   National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2017 [Oral Presentation]
   Abhijeet Kumar, Anjali Anil Shenoy and Avinash Shrama.
   CVIT, KCIS, International Institute of Information Technology, Hyderabad

2. **Towards View-Invariant Intersection Recognition from Videos using Deep Network Ensembles** [Short Oral Presentation] [52]
   International Conference on Intelligent Robots and Systems (IROS), 2018
   Abhijeet Kumar[†], Gunshi Gupta[†], Avinash Sharma and K. Madhav Krishna
   CVIT, RRC, KCIS, International Institute of Information Technology, Hyderabad

[†] Authors contributed Equally

Bibliography

[1] Gardens Point Dataset. https://wiki.qut.edu.au/display/cyphy/Day+and+Night+with+Lateral+Pose+Change+Datasets.

[2] Grand Theft Auto V. https://en.wikipedia.org/wiki/Development_of_Grand_Theft_Auto_V.

[3] Grand Theft Auto V Auto-drive Mod. https://www.gta5-mods.com/scripts/vautodrive.

[4] Mapillary. https://www.mapillary.com/app.

[5] S. Achar, C. V. Jawahar, and K. M. Krishna. Large scale visual localization in urban environments. In *ICRA*, 2011.

[6] A. Angeli, D. Filliat, S. Doncieux, and J. Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *TRO*, 2008.

[7] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[8] K. Barnard, P. Duygulu, D. Forsyth, N. d. Freitas, D. M. Blei, and M. I. Jordan. Matching words and pictures. *Journal of machine learning research*, 3(Feb):1107–1135, 2003.

[9] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *TPAMI*, 2013.

[10] D. Bhatt, D. Sodhi, A. Pal, V. Balasubramanian, and K. M. Krishna. Have I reached the intersection: A deep learning-based approach for intersection detection from monocular cameras. In *IROS*, 2017.

[11] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann. Netgan: Generating graphs via random walks. *arXiv preprint arXiv:1803.00816*, 2018.

[12] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *CoRR*, abs/1611.08097, 2016.

[13] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[14] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 29(3):394–410, 2007.

[15] H. Cevikalp, B. Benligiray, O. N. Gerek, and H. Saribas. Semi-supervised robust deep neural networks for multi-label classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 9–17, 2019.

[16] J. Chen, T. Ma, and C. Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

[17] Z. Chen, A. Jacobson, N. Sunderhauf, B. Upcroft, L. Liu, C. Shen, I. Reid, and M. Milford. Deep learning features at scale for visual place recognition. *arXiv preprint arXiv:1701.05105*, 2017.

[18] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo. Multi-label image recognition with graph convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5177–5186, 2019.

[19] F. Chung. *Spectral Graph Theory*. Number no. 92 in CBMS Regional Conference Series. Conference Board of the Mathematical Sciences.

[20] S. Datta, S. Tourani, A. Sharma, and K. M. Krishna. Slam pose-graph robustification via multi-scale heat-kernel analysis. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 2912–2919. IEEE, 2016.

[21] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[22] T. Derr, Y. Ma, and J. Tang. Signed graph convolutional networks. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 929–934. IEEE, 2018.

[23] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *CoRL*, 2017.

[24] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[25] P. Duygulu, K. Barnard, J. F. de Freitas, and D. A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *European conference on computer vision*, pages 97–112. Springer, 2002.

[26] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[27] S. Feng, R. Manmatha, and V. Lavrenko. Multiple bernoulli relevance models for image and video annotation. In *null*, pages 1002–1009. IEEE, 2004.

[28] H. Fu, Q. Zhang, and G. Qiu. Random forest for image annotation. In *European Conference on Computer Vision*, pages 86–99. Springer, 2012.

[29] D. Gálvez-López and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *TRO*, 2012.

[30] H. Gao, Z. Wang, and S. Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1416–1424. ACM, 2018.

[31] A. Gawel, C. Del Don, R. Siegwart, J. Nieto, and C. Cadena. X-view: Graph-based semantic multi-view localization. *arXiv preprint arXiv:1709.09905*, 2017.

[32] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth. OpenFABMAP: An open source toolbox for appearance-based loop closure detection. In *ICRA*, 2012.

[33] D. Grangier and S. Bengio. A discriminative kernel-based model to rank images from text queries. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 10(LIDIAP-ARTICLE-2008-010), 2008.

[34] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 309–316. IEEE, 2009.

[35] H. Guo, K. Zheng, X. Fan, H. Yu, and S. Wang. Visual attention consistency under image transforms for multi-label image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[36] A. A. Hafez, M. Singh, K. M. Krishna, and C. V. Jawahar. Visual localization in highly crowded urban environments. In *IROS*, 2013.

[37] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[38] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[39] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[40] H. Hu, G.-T. Zhou, Z. Deng, Z. Liao, and G. Mori. Learning structured inference neural networks with label relations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2960–2968, 2016.

[41] M. J. Huiskes and M. S. Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 39–43. ACM, 2008.

[42] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 119–126. ACM, 2003.

[43] J. Jin and H. Nakayama. Annotation order matters: Recurrent image annotator for arbitrary length image tagging. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2452–2457. IEEE, 2016.

[44] J. Johnson, L. Ballan, and L. Fei-Fei. Love thy neighbors: Image annotation by exploiting image metadata. In *Proceedings of the IEEE international conference on computer vision*, pages 4624–4632, 2015.

[45] M. M. Kalayeh, H. Idrees, and M. Shah. Nmf-knn: Image annotation using weighted multi-view non-negative matrix factorization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 184–191, 2014.

[46] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[47] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[48] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[49] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML-W*, 2015.

[50] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[51] A. Kumar, A. A. Shenoy, and A. Sharma. Mkl based local label diffusion for automatic image annotation. *Sixth National Conference on Computer Vision, Pattern Recognition, Image Processing, and Graphics. NCVPRIPG 2017*, 841:385–399, 2017.

[52] A. Kumar, A. A. Shenoy, and A. Sharma. Towards view-invariant intersection recognition from videos using deep network ensembles. *IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS 2018*, 2018.

[53] V. Lavrenko, R. Manmatha, and J. Jeon. A model for learning the semantics of pictures. In *Advances in neural information processing systems*, pages 553–560, 2004.

[54] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[55] Q. Li, L. Chen, Q. Zhu, M. Li, Q. Zhang, and S. S. Ge. Intersection detection and recognition for autonomous urban driving using a virtual cylindrical scanner. *IET Intelligent Transport Systems*, 2013.

[56] R. Li, S. Wang, F. Zhu, and J. Huang. Adaptive graph convolutional neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[57] X. Li, C. G. Snoek, and M. Worring. Learning social tag relevance by neighbor voting. *IEEE Transactions on Multimedia*, 11(7):1310–1322, 2009.

[58] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[59] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[60] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[61] C. Linegar, W. Churchill, and P. Newman. Made to measure: Bespoke landmarks for 24-hour, all-weather localisation with a camera. In *ICRA*, 2016.

[62] F. Liu, T. Xiang, T. M. Hospedales, W. Yang, and C. Sun. Semantic regularisation for recurrent image annotation. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 4160–4168. IEEE, 2017.

[63] J. Liu, M. Li, Q. Liu, H. Lu, and S. Ma. Image annotation via graph learning. *Pattern recognition*, 42(2):218–228, 2009.

[64] Y. Luo, M. Jiang, and Q. Zhao. Visual attention in multi-label image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[65] A. Makadia, V. Pavlovic, and S. Kumar. Baselines for image annotation. *International Journal of Computer Vision*, 90(1):88–105, 2010.

[66] K. Marino, R. Salakhutdinov, and A. Gupta. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844*, 2016.

[67] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[68] M. J. Milford and G. F. Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *ICRA*, pages 1643–1649, May 2012.

[69] M. J. Milford and G. F. Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *ICRA*, 2012.

[70] F. Monay and D. Gatica-Perez. Plsa-based image auto-annotation: constraining the latent space. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 348–351. ACM, 2004.

[71] V. N. Murthy, E. F. Can, and R. Manmatha. A hybrid model for automatic image annotation. In *Proceedings of International Conference on Multimedia Retrieval*, page 369. ACM, 2014.

[72] V. N. Murthy, S. Maji, and R. Manmatha. Automatic image annotation using deep learning representations. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 603–606. ACM, 2015.

[73] V. N. Murthy, A. Sharma, V. Chari, and R. Manmatha. Image annotation using multi-scale hypergraph heat diffusion framework. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 299–303. ACM, 2016.

[74] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017.

[75] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016.

[76] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*, 2018.

[77] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[78] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016.

[79] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez. The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016.

[80] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[81] J. L. Schönberger, M. Pollefeys, A. Geiger, and T. Sattler. Semantic visual localization. *arXiv preprint arXiv:1712.05773*, 2017.

[82] M. Schuster, K. K. Paliwal, and A. General. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 1997.

[83] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN Features off-the-shelf: an astounding baseline for recognition. In *CVPR workshops*, 2014.

[84] A. Sharma, R. Horaud, J. Cech, and E. Boyer. Topologically-robust 3d shape matching based on diffusion geometry and seed growing. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2481–2488. IEEE, 2011.

[85] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *arXiv preprint arXiv:1211.0053*, 2012.

[86] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[87] N. Sunderhauf, S. Shirazi, A. Jacobson, F. Dayoub, E. Pepperell, B. Upcroft, and M. Milford. Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. In *RSS*, 2015.

[88] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[89] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[90] A. D. Szlam, M. Maggioni, and R. R. Coifman. Regularization on graphs with function-adapted diffusion processes. *Journal of Machine Learning Research*, 9(Aug):1711–1739, 2008.

[91] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf.

[92] T. Uricchio, L. Ballan, L. Seidenari, and A. Del Bimbo. Automatic image annotation via label transfer in the semantic space. *Pattern Recognition*, 71:144–157, 2017.

[93] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[94] J. Verbeek, M. Guillaumin, T. Mensink, and C. Schmid. Image annotation with tagprop on the mirflickr set. In *Proceedings of the international conference on Multimedia information retrieval*, pages 537–546. ACM, 2010.

[95] Y. Verma and C. Jawahar. Exploring svm for image annotation in presence of confusing labels. In *BMVC*, pages 25–1, 2013.

[96] Y. Verma and C. Jawahar. Image annotation by propagating labels from semantic neighbourhoods. *International Journal of Computer Vision*, 121(1):126–148, 2017.

[97] H. Wang, H. Huang, and C. Ding. Image annotation using bi-relational graph of images and semantic labels. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 793–800. IEEE, 2011.

[98] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu. Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2285–2294, 2016.

[99] Z. Wang, T. Chen, G. Li, R. Xu, and L. Lin. Multi-label image recognition by recurrently discovering attentional regions. In *Proceedings of the IEEE international conference on computer vision*, pages 464–472, 2017.

[100] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[101] Y. Xiang, X. Zhou, T.-S. Chua, and C.-W. Ngo. A revisit of generative model for automatic image annotation using markov random fields. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1153–1160. IEEE, 2009.

[102] M. Yeh and Y. Li. Multilabel deep visual-semantic embedding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[103] W. Yih, K. Toutanova, J. C. Platt, and C. Meek. Learning discriminative projections for text similarity measures. In *CoNLL*, 2011.

[104] W. Yu, C. Zheng, W. Cheng, C. C. Aggarwal, D. Song, B. Zong, H. Chen, and W. Wang. Learning deep network representations with adversarially regularized autoencoders. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2663–2671. ACM, 2018.

[105] H. Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2126–2136. IEEE, 2006.

[106] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*, 2018.

[107] J. Zhang, Q. Wu, C. Shen, J. Zhang, and J. Lu. Multilabel image classification with regional latent semantic dependencies. *IEEE Transactions on Multimedia*, 20(10):2801–2813, 2018.

[108] Q. Zhu, Q. Mao, L. Chen, M. Li, and Q. Li. Veloregistration based intersection detection for autonomous driving in challenging urban scenarios. In *ITSC*, 2012.