National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics
Speech and Language Processing Group

# Graph Neural Networks with External Knowledge for Visual Dialog

Diploma Thesis

of

**IOANNIS KALOGEROPOULOS**

**Supervisor:** Alexandros Potamianos
Associate Professor, NTUA

Athens, July 2022

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics
Speech and Language Processing Group

# Graph Neural Networks with External Knowledge for Visual Dialog

Diploma Thesis

of

**IOANNIS KALOGEROPOULOS**

**Supervisor:** Alexandros Potamianos
Associate Professor, NTUA

Approved by the examination committee on 21th July 2022.

| | | |
|---|---|---|
| *(Signature)* | *(Signature)* | *(Signature)* |
| ............................ | ............................ | .................. |
| Alexandros Potamianos | Constantinos Tzafestas | Stefanos Kollias |
| Associate Professor, NTUA | Associate Professor, NTUA | Professor, NTUA |

Athens, July 2022

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics
Speech and Language Processing Group

*(Signature)*

. . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Ioannis Kalogeropoulos

Electrical & Computer
Engineer

# Περίληψη

Στην παρούσα Διπλωματική Εργασία μελετήθηκε η αναπαράσταση δεδομένων ως γράφων και η κωδικοποίησή τους με Βαθιά Νευρωνικά Δίκτυα, με σκοπό τη χρήση τους για το πρόβλημα του Οπτικού Διαλόγου. Για την επίτευξη ανταγωνιστικών αποτελεσμάτων έγιναν πειράματα προς δύο κατευθύνσεις:

- **Μέθοδοι Σύμμειξης Τροπικοτήτων.** Για την επίλυση ενός μεγάλου συνόλου προβλημάτων που συνενώνουν τους τομείς της Επεξεργασίας Φυσικής Γλώσσας και της Όρασης Υπολογιστών, η μέθοδος που θα χρησιμοποιηθεί για τη συγχώνευση των διαφόρων ειδών δεδομένων του προβλήματος αποτελεί μία από τις βασικότερες σχεδιαστικές επιλογές στην κατασκευή του μοντέλου και έναν από τους καίριους παράγοντες που δύναται να οδηγήσουν σε καλύτερα αποτελέσματα. Για την περίπτωση του Οπτικού Διαλόγου, πραγματοποιήθηκαν πειράματα ως προς τη σύντηξη των δύο διαφορετικών τροπικοτήτων , εικόνας και κειμένου.
- **Ενσωμάτωση Εξωτερικής Γνώσης.** Το πρόβλημα του οπτικού διαλόγου δεν απαιτεί εκ των πραγμάτων κάποια εξωτερική γνώση. Ωστόσο, η εισαγωγή εξωτερική γνώσης τείνει να οδηγεί σε καλύτερα αποτελέσματα σε πολλά προβλήματα στο χώρο της Μηχανικής Μάθησης και ακόμα περισσότερο σε προβλήματα που ανήκουν στο χώρο της Επεξεργασίας Φυσικής Γλώσσας. Για το λόγο αυτό μελετάται συχνά από ερευνητές η επιστράτευσή της σε πολλές διαφορετικές εφαρμογές και στην παρούσα εργασία επιχειρούμε και εμείς να εκμεταλλευτούμε την επιπλέον πληροφορία που θα μας προσφέρει.

Στα πειράματά μας χρησιμοποιούμε τις μεθόδους σύμμειξης τροπικοτήτων του αρχικού μας μοντέλου, προκειμένου να συνδυάσουμε τις τρεις τροπικότητες του μοντέλου μας. Στη συνέχεια, πειραματιζόμαστε με την κωδικοποίηση της εξωτερικής γνώσης. Συγκεκριμένα, ερευνάμε τη χρήση ενός ή πολλαπλών τύπων ακμών του γράφου γνώσης και μελετάμε διαφορετικούς τρόπους συλλογής της εξωτερικής γνώσης.

Πραγματοποιώντας ένα σύνολο από πειράματα μπορούμε να εξάγουμε συμπεράσματα για την επίδραση της εισαγωγής εξωτερικής γνώσης στο μοντέλο. Συγκεκριμένα, ξεπερνώντας με δύο διαφορετικές μεθόδους τα αποτελέσματα του υλοποιημένου από εμάς αρχικού μοντέλου, αποδεικνύουμε την ευεργετική επίδραση αυτής στη συνολική επίδοση του μοντέλου. Επιπρόσθετα, αποδεικνύουμε την επίδραση αυτή πραγματοποιώντας πειράματα χρησιμοποιώντας δύο διαφορετικούς αποκωδικοποιητές. Η συνέπεια στα αποτελέσματα των μοντέλων χρησιμοποιώντας και τους δύο αποκωδικοποιητές τονίζει την επίδραση των διαφορετικών κωδικοποιητών στη συνολική απόδοση του μοντέλου.

**Λέξεις Κλειδιά**
Βαθιά Μάθηση, Επεξεργασία Φυσικής Γλώσσας, Οπτικός Διάλογος Graph Neural Networks, Knowledge Graphs, Multimodal Fusion,

# Abstract

In this Diploma Thesis, we study the effectiveness of Graph Neural Networks on the task of Visual Dialog. Towards achieving interesting architectures and great results, we experiment on two axes:

- **Fusion Methods.** In a wide range of Machine Learning problems, we encounter the problem of combining different types of information extracted from various sources. The fusion method used to combine the different modalities is a fundamental design choice of the model and a crucial factor towards the achievement of better results. We experimented on a few sets of different methods and selected the best one for our model.

- **External Knowledge** The task of Visual Dialog doesn't require by itself the use of external knowledge. Nevertheless, introducing external knowledge has been proved effective in many tasks of Machine Learning and especially in the field of Natural Language Processing. As a result it has drawn a lot research interest through the last years and has been applied to a wide variety of similar tasks. Hence we attempt to introduce external knowledge to our approach and experiment with a few ways of exploiting the extra information.

In our experiments we adapt the fusion methods of our baseline and utilize them for fusing the three modalities of our model. We further experiment on the encoding of the External Knowledge. Specifically, we examine the use of one or multiple types of relations of the knowledge graph as well as different methods of aggregating the external information.

By conducting a number of experiments, we are able to draw interesting conclusions about the impact of introducing External Knowledge to our model. Specifically, by surpassing the implemented baseline using two different methods, we conclude that it is beneficial for the overall performance. Moreover, we demonstrate this impact by using two types of decoders. The consistency of the results using both decoders highlights the impact of the different encoders. Finally, from our results, we come to the conclusion that the simplest models with less parameters were able to perform better towards encoding the External Knowledge Graph.

## Keywords

Deep Learning, Natural Language Processing, Graph Neural Networks, Knowledge Graphs, Visual Dialog

# Ευχαριστίες

Στις πρώτες σελίδες αυτής της διπλωματικής εργασίας θέλω να ευχαριστήσω και να εκφράσω την ευγνωμοσύνη μου σε ορισμένους ανθρώπους, που ο καθένας με τον τρόπο του υπήρξε αρωγός τόσο στην εκπόνηση αυτής, όσο και στη συνολικότερη εξέλιξή μου κατά τη διάρκεια των φοιτητικών μου χρόνων, ενός σημαντικού κεφαλαίου της ζωής μου που ολοκληρώνεται μέσα από αυτήν την εργασία.

Αναμφίβολα, το μεγαλύτερο ευχαριστώ το οφείλω στον επιβλέποντα καθηγητή της παρούσας μελέτης, τον καθηγητή Αλέξανδρο Ποταμιάνο. Μέσα από τα μαθήματά του με μύησε στο αντικείμενό του και στον κόσμο της έρευνας, ενώ μέσα από τη συνεργασία μας όλους αυτούς του μήνες μου χάρισε πλήθος ερεθισμάτων γνώσης και μάθησης. Οι πολύπλευρες γνώσεις του και οι καίριες συμβουλές του υπήρξαν καθοριστικής σημασίας.

Στη συνέχεια, θέλω να ευχαριστήσω ιδιαίτερα και τους διδακτορικούς ερευνητές της ομάδας του κυρίου Ποταμιάνου και ειδικότερα τον Ευθύμη Γεωργίου, ο οποίος είχε πάντα τη διάθεση να με κατευθύνει και του οποίου η συμβολή ήταν καθοριστική για την εκπόνηση της παρούσας διπλωματικής εργασίας.

Φυσικά, οφείλω ένα μεγάλο ευχαριστώ την οικογένειά μου, τη μητέρα μου και την αδερφή μου. Χωρίς την αμέριστη στήριξή τους όλα αυτά τα χρόνια, δε θα τα είχα καταφέρει.

Τέλος, θέλω να ευχαριστήσω βαθιά όλους τους φίλους μου που ήταν διαθέσιμοι για κάθε είδους βοήθεια καθ'όλη τη διάρκεια των σπουδών μου και των οποίων η παρουσία σημάδεψε αυτό το κεφάλαιο της ζωής μου.

Καλογερόπουλος Ιωάννης

Αθήνα, Ιούλιος 2022

# Contents

**Περίληψη** **1**

**Abstract** **3**

**Ευχαριστίες** **7**

**List of Figures** **13**

**List of Tables** **15**

**0 Εκτεταμένη Ελληνική Περίληψη** **17**
- 0.1 Εισαγωγή . . . 17
- 0.2 Θεωρητικό Υπόβαθρο . . . 18
  - 0.2.1 Νευρωνικά Δίκτυα Γράφων . . . 18
- 0.3 Σχετική Βιβλιογραφία . . . 18
- 0.4 Διατύπωση Προβλήματος . . . 19
- 0.5 Προτεινόμενο Μοντέλο . . . 19
- 0.6 Πειράματα . . . 25
  - 0.6.1 Διακρίνων αποκωδικοποιητής . . . 25
  - 0.6.2 Γεννητικός Αποκωδικοποιητής . . . 26
- 0.7 Σχολιασμός αποτελεσμάτων . . . 27
- 0.8 Συμπεράσματα . . . 27

**1 Introduction** **29**
- 1.1 What is Artificial Intelligence? . . . 29
- 1.2 Motivation . . . 30
- 1.3 Research Contributions . . . 30
- 1.4 Thesis Outline . . . 30

**2 Machine Learning** **33**
- 2.1 Introduction . . . 33
- 2.2 Learning Categories . . . 34
  - 2.2.1 Supervised Learning . . . 34
  - 2.2.2 Unsupervised Learning . . . 35
  - 2.2.3 Semi-Supervised Learning . . . 35
- 2.3 Supervised Machine Learning Algorithms . . . 35
  - 2.3.1 Activation Function . . . 35
  - 2.3.2 Loss Function . . . 37
  - 2.3.3 Gradient descent . . . 39
  - 2.3.4 Backpropagation . . . 40
  - 2.3.5 Generalization . . . 40

2.3.6 Tackling overfitting . . . . . 41
2.3.7 Machine Learning Models . . . . . 41
2.4 Deep Learning Models . . . . . 42
2.4.1 Feedforward Neural Networks . . . . . 43
2.4.2 Convolutional Neural Networks (CNN) . . . . . 44
2.4.3 Recurrent Neural Networks . . . . . 45

**3 Graph Neural Networks 49**
3.1 Introduction . . . . . 49
3.2 Characteristics of GNNs . . . . . 50
3.2.1 Permutation invariance and equivariance . . . . . 50
3.2.2 Locality on graphs . . . . . 51
3.3 Models . . . . . 52
3.3.1 Graph Neural Networks . . . . . 52
3.3.2 General Graph Neural Network Frameworks . . . . . 53
3.3.3 Variants of Graph Neural Networks . . . . . 56
3.3.4 How to use GNNs? . . . . . 61
3.3.5 Knowledge Graphs - ConceptNet . . . . . 61

**4 Introduction to NLP and Visual Dialog 65**
4.1 What is Visual Dialog? . . . . . 65
4.2 Natural Language Processing . . . . . 66
4.2.1 Applications . . . . . 67
4.2.2 Earlier Approaches . . . . . 67
4.2.3 Word Embeddings . . . . . 68
4.2.4 Denotational Representation . . . . . 68
4.2.5 Distributional Semantics . . . . . 68
4.2.6 Embeddings from Language Models (ELMo) . . . . . 70
4.3 Computer Vision . . . . . 70
4.3.1 Faster R-CNN . . . . . 71

**5 Proposed Model 73**
5.1 Introduction . . . . . 73
5.2 Dataset . . . . . 73
5.3 Related Work . . . . . 76
5.4 Baseline model . . . . . 77
5.5 Proposed Method . . . . . 81
5.5.1 Core Architecture . . . . . 81
5.5.2 GNNs for external Graph . . . . . 85
5.6 Experiments . . . . . 88
5.6.1 Discriminative Decoder . . . . . 88
5.6.2 Generative Decoder . . . . . 89
5.6.3 Results Discussion . . . . . 89

**6 Conclusions 91**
6.1 Discussion . . . . . 91
6.2 Future Work . . . . . 92

**Bibliography 95**

**Bibliography** **101**

**List of Abbreviations** **103**

**7 Appendix** **105**

7.1 Implementation Details . . . . . 105

7.2 Preprocessing . . . . . 105

7.2.1 Dataset preprocessing . . . . . 105

7.2.2 Preprocess of ConceptNet . . . . . 105

# List of Figures

1 Συνοπτική απεικόνιση του προτεινόμενου μοντέλου . . . . . . . . . . . . . . . . . . 20
2 Η αρχιτεκτονική του προτεινόμενου μοντέλου . . . . . . . . . . . . . . . . . . . . . 21
3 Ενημέρωση των κεντρικών κόμβων $V^1$ (καφε) αθροίζοντας ολόυς τους γύρω κόμβους $V^2$ (μπλε). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
4 Εφαρμογή μηχανισμού πύλης αρχικά σε επίπεδο κόμβων και στη συνέχεια σε επίπεδο γράφων. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
5 Εφαρμογή μηχανισμών πυλών για τον υπολογισμό των συνολικών γνώσεων για την κάθε τροπικότητα και στη συνέχεια για τον συνδυασμό αυτων. . . . . . . . . . . . . 25

2.1 Artificial Intelligence, Machine Learning and Deep Learning Domains . . . . . . . 33
2.2 Types of Learning . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
2.3 The sigmoid function . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
2.4 The tanh function . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
2.5 The ReLU function . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
2.6 The Leaky ReLU function . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
2.7 The ELU function . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
2.8 Classification of data by support vector machine (SVM). Source: [1] . . . . . . . . 43
2.9 Typical CNN architecture. Source: researchgate.net . . . . . . . . . . . . . . . . 45
2.10 A recurrent neural network unfolded for a sequence of length t. Source: datascience.eu . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
2.11 The repeating module in an LSTM. Source: colah.github.io . . . . . . . . . . . . 47

3.1 Some applications where the information is represented by graphs: (a) a chemical compound (adrenaline), (b) an image, and (c) a subset of the web. Source: [2] . . 49
3.2 Permutation invariance . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
3.3 Updates in a GN block. Blue indicates the element that is being updated, and black indicates other elements which are involved in the update. Source: [3] . . . 56
3.4 **Left**: The attention mechanism $a(W\overrightarrow{h}_i, W\overrightarrow{h}_j)$ employed by GAT, parametrized by a weight vector $\overrightarrow{a}\epsilon\mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right**: An illustration of multihead attention (with K = 3 heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain h1. Source: [4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59
3.5 Diagram for computing the update of a single graph node/entity (red) in the R-GCN model. Activations (d-dimensional vectors) from neighboring nodes (dark blue) are gathered and then transformed for each relation type individually (for both in- and outgoing edges). The resulting representation (green) is accumulated in a (normalized) sum and passed through an activation function. This per-node update can be computed in parallel with shared parameters across the whole graph. Source: [5] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60

3.6 Common use cases of GNNs . . . . . . . . . . . . . . . . . . . . . . . . . . 61
3.7 Application fields of Knowledge Graphs. Source: [6] . . . . . . . . . . . . . . . . 62
3.8 An example sub-graph of a Knowledge Graph. Source: [7] . . . . . . . . . . . . . 62

4.1 An example dialog of the Visual Dialog task. Source: visualdialog.org . . . . . . . 66
4.2 The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. Source: [8] . . . . 70
4.3 An overview of the Faster R-CNN architecture. The RPN module serves as the ‘attention’ of the unified network. Source: [9] . . . . . . . . . . . . . . . . . . . . 71

5.1 Examples of COCO images [10] . . . . . . . . . . . . . . . . . . . . . . . . . . 74
5.2 Differences between image captioning, Visual Question Answering (VQA) and Visual Dialog [11] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
5.3 Our main baseline: Knowledge-Bridge Graph Network [12] . . . . . . . . . . . . 78
5.4 Knowledge Encoding module in Knowledge-Bridge Graph Network [12] . . . . . . 78
5.5 Knowledge Storage’s steps **2** and **3**, where the edges between a vision node (blue) and the text nodes (green) are updated and then the each vision node is updated through a Query-Guided Cross Graph Convolution resulting in the updated visual nodes (purple). Source: [12] . . . . . . . . . . . . . . . . . . . . . . . . . . 79
5.6 Knowledge Storage’s steps **4**, **5**, **6** and **7**, where the "G" represents a gate operation. Source: [12] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
5.7 Knowledge Retrieval module with multi-type decoders. “G” represents gate operation. Source: [12] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
5.8 Abstract visualisation of the proposed model . . . . . . . . . . . . . . . . . . . . 81
5.9 The proposed model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 82
5.10 Update of the center nodes $V_1$ (brown) by summing all the cross nodes $V_2$ (blue). 84
5.11 Apply gate operations firstly one node level and then on graph level to acquire the final enriched information. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85
5.12 Apply gate operations on the two types of knowledge for each modality and then one final gate operation between the two modalities to extract the final embedding. 86
5.13 GCN for encoding the External Knowledge Graph. . . . . . . . . . . . . . . . . . 86
5.14 R-GCN for encoding the External Knowledge Graph. . . . . . . . . . . . . . . . . 87

7.1 Blue nodes represent the original nodes, green the nodes that are selected for the $V^e$ set, while in the red ones the nodes that are not. . . . . . . . . . . . . . . . . 106
7.2 Only the green node will be added to $V^e$ and the red ones will be discarded. . . . 107
7.3 Frequency of relations in train split. . . . . . . . . . . . . . . . . . . . . . . . . 110
7.4 Frequency of relations in val split. . . . . . . . . . . . . . . . . . . . . . . . . . 110
7.5 Frequency of relations in test split. . . . . . . . . . . . . . . . . . . . . . . . . 111

# List of Tables

1 Αποτελέσματα στο test split χρησιμοποιώντας τον Διακρίνων αποκωδικοποιητή. . . 26

2 Αποτελέσματα στο val split χρησιμοποιώντας τον Διακρίνων αποκωδικοποιητής. . . 26

3 Αποτελέσματα στο val split χρησιμοποιώντας τον Γεννητικό αποκωδικοποιητή. . . . 27

5.1 Common points between Visual Dialog and Visual Question Answering . . . . . . 75

5.2 Differences between Visual Dialog and Visual Question Answering . . . . . . . . . 75

5.3 Results of baseline and various successful models on test-standard set of VisDial v1.0 on discriminative method. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80

5.4 Result comparison on test-standard set of VisDial v1.0 on discriminative method. 89

5.5 Result comparison on validation set of VisDial v1.0 on discriminative method. . . 89

5.6 Result comparison on val-standard set of VisDial v1.0 on generative method. . . . 90

7.2 The relation types used for creating the subgraphs of ConceptNet. . . . . . . . . . 109

# Chapter 0

# Εκτεταμένη Ελληνική Περίληψη

## 0.1 Εισαγωγή

Στην παρούσα διπλωματική ασχολούμαστε με το πρόβλημα του Οπτικού Διαλόγου (Visual Dialog). Το πρόβλημα αυτό διατέμνει τα πεδία της Επεξεργασίας Φυσικής Γλώσσας και της Όρασης Υπολογιστών, πεδία που έχουν γνωρίσει μεγάλη ανάπτυξη τα τελευταία χρόνια και έχουν εφαρμογές σε πληθώρα άλλων προβλημάτων. Το πρόβλημα του Οπτικού Διαλόγου προϋποθέτει έναν πράκτορα (AI agent) να συνομιλήσει με ένα χρήστη σε φυσική γλώσσα σχετικά με ένα οπτικό αντικείμενο. Πιο συγκεκριμένα, ο σκοπός μας είναι παρέχοντας μία εικόνα και μία περιγραφή αυτής σε ένα μοντέλο, το μοντέλο αυτό να είναι ικανό να πραγματοποιεί ουσιαστικό διάλογο με τον χρήστη. Ο χρήστης παραθέτει ερωτήσεις στο μοντέλο, το οποίο καλείται να συνδυάσει την τρέχουσα ερώτηση με το ιστορικό του διαλόγου, να εντοπίσει στην εικόνα τα αντικείμενα τα οποία αφορά η τρέχουσα ερώτηση και να καταλήξει σε μία κατάλληλη απάντηση.

Ο διάλογος ενός χρήστη με έναν υπολογιστή είναι ένα πρόβλημα που έχει προσεγγιστεί από αρκετούς ερευνητές τα τελευταία χρόνια και αποκτά ολοένα και περισσότερη δημοφιλία εξαιτίας της πληθώρας των εφαρμογών του. Παρ' όλα αυτά το μεγαλύτερο κομμάτι αυτής της έρευνας έχει εστιάσει σε γλωσσικούς μόνο διαλόγους, όπου ο χρήστης και ο υπολογιστής συμμετέχουν σε μία "απλή" συνομιλία, επιτυγχάνοντας εντυπωσιακά αποτελέσματα όπως οι περίφημοι εικονικοί βοηθοί Alexa, Siri κ.α.. Η εισαγωγή μίας εικόνας και η ανάπτυξη διαλόγου γύρω από αυτή αποτελεί μία ενδιαφέρουσα επέκταση του κλασσικού διαλόγου η οποία δύναται να έχει πολλές ενδιαφέρουσες εφαρμογές, όπως η διευκόλυνση ατόμων με προβλήματα όρασης στο να αντιληφθούν καλύτερα το περιβάλλον τους ή μία κατάσταση που απεικονίζεται σε μία εικόνα και η ανάπτυξη πιο χρήσιμων και ελκυστικών εικονικών βοηθών.

Η προσέγγιση του προβλήματος στην παρούσα εργασία αποτελεί μία πολυεπίπεδη διεργασία, η οποία πραγματοποιείται με χρήση τεχνολογιών αιχμής από τους τομεις της μηχανικής μάθησης και των νευρωνικών δικτύων. Η αναπαράσταση των στοιχείων διαλόγου γίνεται σε επίπεδο γράφων, τόσο για τους γύρους του διαλόγου όσο και για τα αντικείμενα της εικόνας. Επιπρόσθετα, πειραματιζόμαστε με την εισαγωγή εξωτερικής γνώσης, την οποία αναπαριστάμε επίσης χρησιμοποιώντας έναν ακόμα γράφο. Επιλέγοντας ως αρχικό μοντέλο μία προσέγγιση που επιτυγχάνει κορυφαία αποτελέσματα και χρησιμοποιεί προτότυπη αρχιτεκτονική, το επεκτείνουμε δοκιμάζοντας την ενσωμάτωση της εξωτερικής γνώσης σε αυτό και ποικίλους τρόπους για το συνδυασμό των διαφορετικών πηγών πληροφορίας, εικόνα, διάλογος και εξωτερική γνώση. Τα αποτελέσματά μας υποδεικνύουν ότι η ενσωμάτωση της εξωτερικής γνώσης επιτρέπει το μοντέλο να επιτύχει καλύτερα αποτελέσματα, βοηθώντας το να αντιληφθεί καλύτερα τη γενική έννοια του κάθε αντικειμένου που αναφέρθηκε στον διάλογο.

## 0.2 Θεωρητικό Υπόβαθρο

Σε αυτήν την ενότητα θα κάνουμε μία συνοπτική επισκόπηση των κύριων τεχνολογιών που χρησιμοποιούνται στην παρούσα εργασία.

### 0.2.1 Νευρωνικά Δίκτυα Γράφων

Οι γράφοι είναι ένα τύπος δομής δεδομένων που χρησιμοποιείται για την αναπαράσταση αντικειμένων μέσω κόμβων και ακμών. Τα τελευταία χρόνια η ανάλυση γράφων μέσω της μηχανικής μάθησης έχει κεντρίσει ιδιαίτερα το ενδιαφέρον της επιστημονική κοινότητας, κυρίως λόγω της εκφραστικής δύναμης των γράφων, αλλά και της πληθώρας των τομέων στους οποίους μπορούν να εφαρμοστούν, όπως η ανάλυση κοινωνικών δικτύων [13], [14], οι φυσικές επιστήμες [15], [16], η μοντελοποίηση και η αλληλεπιδραση μεταξύ των προτεϊνών [17]), η αναπαράσταση γνώσης [18] και πολλοί άλλοι τομείς [19].

Βασισμένα στα συνελικτικά νευρωνικά δίκτυα, τα νευρωνικά δίκτυα γράφων μοιράζονται με αυτά αρκετά κοινά σημεία, όπως την τοπικότητα, τα κοινά βάρη και τη χρήση πολλαπλών επιπέδων, έχουν ωστόσο και ένα ιδιαίτερο χαρακτηριστικό το οποίο τα ξεχωρίζει από τα υπόλοιπα είδη νευρωνικών δικτύων. Συγκεκριμένα, νευρωνικά δίκτυα γράφων συλλέγουν πληροφορία από έναν γράφο, με τέτοιο τρόπο έτσι ώστε το αποτέλεσμα να είναι ανεξάρτητο της διάταξης των κόμβων μέσα σε αυτόν. Η ιδιότητα αυτή σε συνδυασμό με την ικανότητα αυτών των δικτύων να αναπαριστούν την εξάρτηση μεταξύ δύο γειτονικών κόμβων, τα καθιστά ιδανικά για πληθώρα προβλημάτων της μηχανικής μάθησης.

Τα τελευταία χρόνια έχουν προταθεί αρκετά μοντέλα νευρωνικών δικτύων γράφων, παρουσιάζοντας σημαντικές διαφορές στον τρόπο που συλλέγεται η πληροφορία από τους διάφορους κόμβους, στη μέθοδο υπολογισμού των ανανεωμένων αναπαραστάσεων των κόμβων, στα διαφορετικά είδη των κόμβων και ακμών που φέρει ο γράφος, ο οποίος παρέχεται ως είσοδος στο δίκτυο και στον τρόπο με τον οποίο ο δίκτυο αυτο χειρίζεται αυτή τη διαφορετικότητα. Σε κάθε περίπτωση, ωστόσο, τα νευρωνικά αυτα δίκτυα ακολουθούν μία κοινή διαδικασία για τον υπολογισμό των αναπραστάσεων των κόμβων, η οποία συνοπτικά μπορεί να περιγραφεί ως: Κάθε κόμβος

1. Κατασκευάζει το "μήνυμα" του για την τρέχουσα κατάστασή του
2. Συλλέγει πληροφορίες ("μηνύματα") απο τους γειτονές του (με τρόπο ανεξάρτητο της διάταξης των γειτώνων του)
3. Υπολογίζει τη νέα του κατάσταση με βάση τις πληροφορίες που έλαβε

Στη συνέχεια, ύστερα από έναν αριθμό επαναλήψεων της παραπάνω πληροφορίας, οι τελικές αναπαραστάσεις των κόμβων μπορούν να χρησιμοποιηθούν είτε για την εξαγωγή μίας συνολικής αναπαράστασης του γράφου, με τρόπο πάλι ανεξάρτητο της διάταξης των κόμβων, είτε για την ταξινόμηση των κόμβων, την πρόβλεψη ακμών είτε για όποια άλλη διαδικασία απαιτεί το πρόβλημα που αντιμετωπίζουμε.

## 0.3 Σχετική Βιβλιογραφία

Όπως προτάθηκε και από τους δημιουργούς του προβλήματος, [11] του Οπτικού Διαλόγου, μία κατάλληλη προσέγγιση στο πρόβλημα είναι με την αρχιτεκτονική κωδικοποιητή - αποκωδικοποιητή. Συγκεκριμένα, ο πρώτος αναλαμβάνει να μετατρέψει την είσοδο του μοντέλου σε ένα διανυσματικό χώρο, μοντελοποιώντας κατάλληλα τις πληροφοριές και ο δεύτερος να μετατρέχει το διάνυσμα σε μία κατάλληλη έξοδο του μοντέλου. Οι [11] πρότειναν 2 είδη αποκωδικοποιητών, έναν ο οποίος αναλαμβάνει να ταξινομήσει ένα σύνολο από πιθανές απαντήσεις στην τρέχουσα ερώτηση και ένας ο οποίος παράγει απαντήσεις επιλέγοντας την πιο πιθανή ακολουθία λέξεων. Αποτέλεσμα αυτού είναι οι περισσότερες προσεγγίσεις στο πρόβλημα του οπτικού διαλόγου να αφορούν το κομμάτι του κωδικοποιητή.

Οι [11] συνόδευσαν το πρόβλημα του Οπτικού Διαλόγου με τρία βασικά μοντέλα: 1) Μία απλή προσέγγιση ξεχωριστής κωδικοποίησης της εικόνας, του διαλόγου και της τρέχουσας ερώτησης και στη συνέχεια ο συνδυασμός αυτών, 2) ένα μοντέλο που διατηρεί ένας είδος μνήμης για την τελευταία ερώτηση, έτσι ώστε μπορεί να ανακτά πολύτιμες πληροφορίες και 3) ένα μοντέλο που πραγματοποιεί ιεραρχική κωδικοποίηση του ιστορικού του διαλόγου. Στη συνέχεια, η έρευνα προχώρησε από άλλους ερευνητές. Αναφορικά, ο [20] πρότεινε μια οπτική ανάλυση συναναφοράς (co-reference), ο [21] συνδύασε ένα μοντέλο για να παράξουν πιθανές απαντήσεις και ένα για να επιλέξουν τη σωστη, ενω χρησιμοποιήσε επίσης μηχανισμό προσοχής πάνω στο ιστορικό του διαλόγου, ο οποίος λάμβανε υπόψην την εικόνα.

Σχετικά με τις προσεγγίσεις που χρησιμοποίησαν νευρωνικά δίκτυα γράφων, ο [22] προσέγγισε το πρόβλημα εστιάζοντας στην κατασκευή μίας δομής του διαλόγου, ενώ οι [23] και [24] επικεντρώθηκαν σε πιο βαθιά αναπαράσταη της εικόνας, την οποία αναπαράστησαν μέσω ενός γράφου. Ακόμα, ο [25] εφάρμοσε μηχανισμό προσοχής πάνω σε κάθε στοιχείο του διαλόγου. Οι παραπάνω προσεγγίσεις διαφέρουν, επίσης, στο πώς μοντελοποιούν το πρόβλημα ως γράφο (μόνο ένας γράφος για όλο το πρόβλημα, ή ένας γράφος για κάθε πηγή πληροφορίας), στο τι επέλεξαν να αναπαριστά ο κάθε κόμβος (τον κάθε γύρο του διαλόγου, ένα αντικείμενο της εικόνας ή σε χαμηλότερο επίπεδο κάθε στοιχείο του διαλόγου λέξη/αντικείμενο [25]). Επιπρόσθετα, διαφέρουν αρκετά στο πώς επέλεξαν να συλλέξουν την πληροφορία από τους γείτονες του κάθε κόμβου και πώς να υπολογίσουν τη νέα του αναπαράσταση. Τέλος, ο [26] πρότεινε τη χρήση εξωτερικής γνώσης στο πρόβλημα του Οπτικού Διαλόγου.

## 0.4 Διατύπωση Προβλήματος

Στο πρόβλημα του οπτικού διαλόγου, ο υπολογιστής καλείται να συνδυάσει την εικόνα $I$, την περιγραφή της $C$, το ιστορικό του διαλόγου $H_t = \{C, (Q_1, A_1), ..., (Q_{t1}, A_{t1})\}$και την τρέχουσα ερώτηση $Q_t$ σε κάθε γύρο $t$. Ο κύριος σκοπός είναι να ταξινομήσει μία λίστα 100 υποψήφιων απαντήσεων $A = \{A_1, A_2, ..., A_{100}\}$ και να επιστρέψει την καλύτερη απάντηση $A_t$ για κάθε ερώτηση $Q_t$.

## 0.5 Προτεινόμενο Μοντέλο

Η κύρια ιδέα της προσέγγισής μας είναι η προσθήκη μία τρίτης πηγής πληροφορίας, της εξωτερικής γνώσης και ο τρόπος συνδυασμού της με τα άλλα δύο είδη πληροφορίας. Συνοπτικά, η όλη διαδικασία φαίνεται στην εικόνα 1 χωρίζεται σε τρία μέρη: κωδικοποίηση εντός της κάθε πληροφορίας, εμπλουτισμός της πληροφορίας από Εικόνα και Διάλογο από τις άλλες πηγές πληροφορίας και ο τελικός συνδυασμός αυτών πριν εισαχθεί το αποτέλεσμα στον αποκωδικοποιητή.

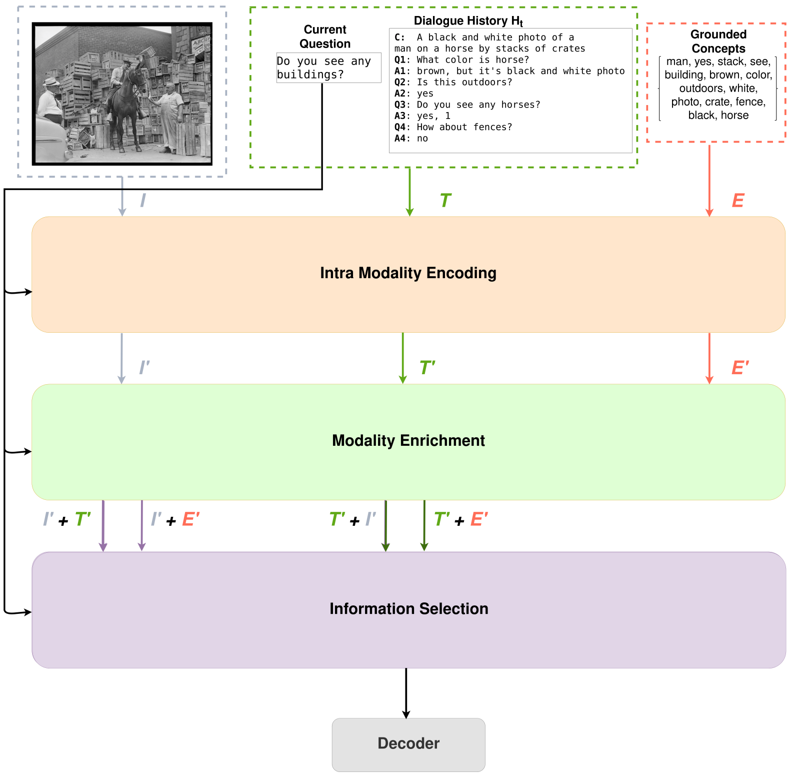

Current Question
Do you see any buildings?
Dialogue History $H_t$
C: A black and white photo of a man on a horse by stacks of crates
Q1: What color is horse?
A1: brown, but it's black and white photo
Q2: Is this outdoors?
A2: yes
Q3: Do you see any horses?
A3: yes, 1
Q4: How about fences?
A4: no
Grounded Concepts
man, yes, stack, see, building, brown, color, outdoors, white, photo, crate, fence, black, horse
I
T
E
Intra Modality Encoding
I'
T'
E'
Modality Enrichment
I' + T'
I' + E'
T' + I'
T' + E'
Information Selection
Decoder

**Figure 1.** *Συνοπτική απεικόνιση του προτεινόμενου μοντέλου*

Πιο αναλυτικά, αρχικά κατασκευάζουμε τρεις γράφους, έναν για κάθε πηγή πληροφορίας. Ο γράφος που αντιστοιχεί στην εικόνα είναι ένας πλήρης γράφος με κόμβους που αντιστοιχούν στα αντικείμενα που υπάρχουν στην εικόνα και ακμές τις μεταξύ τους οπτικές σχέσεις. Ο γράφος που αντιστοιχεί στον διάλογο είναι επίσης ένας πλήρης γράφος με κόμβους την κάθε λέξη του διαλόγου (συμπεριλαμβάνοντας όλους τους γύρους του ιστορικού) και ακμές τη συνένωση μεταξύ των κόμβων. Τέλος, ο γράφος που εισάγει εξωτερική γνώση στο μοντέλο είναι ένας αρκετά αραιός γράφος με πολλάπλά είδη ακμών.

Μία αρκετά πιο αναλυτική απεικόνιση του προτεινόμενου μοντέλου φαίνεται στην εικόνα 2:

Η αρχική κωδικοποίηση εντός της κάθε πηγής πληροφορίας γίνεται ανεξάρτητα και είναι η ίδια διαδικασία για τους τους γράφους της εικόνας και του κειμένου και διαφορετική για τον τρίτο γράφο. Για τους πρώτους δύο αρχικά τροποιούμε την κάθε ακμή εισάγοντας πληροφορία από την τρέχουσα ερώτηση και στη συνέχεια υπολογίζουμε την ανανεωμένη τιμή κάθε κόμβου ως το σταθμισμένο άθροισμα των κόμβων με βάρη που προκύπτουν από τις ακμές. Συμβολίζοντας ως $v$ του κόμβους του κάθε γράφου και ως $e$ τις ακμές, οι παρακάτω εξισώσεις περιγράφουν την κωδικοποίηση:

$$a_{ij} = softmax(W_e(W_1 Q_t \circ W_2 e_{ij})) \tag{1}$$

$$\hat{e}_{ij} = a_{ij} e_{ij} \tag{2}$$

$$\beta_{ij} = softmax(W_v(Q_t \circ W_3[u_j, \hat{e}_{ij}])) \tag{3}$$

$$\hat{v}_i = \sum_{j=1}^{N} \beta_{ij} u_j \tag{4}$$

Για τον γράφο που αναπαριστά την εξωτερική γνώση δοκιμάζουμε μία σειρά από πειράματα, με σκοπό να βρούμε τον καλύτερο τρόπο να τον αξιοποιήσουμε. Συγκεκριμένα, πραγματοποιούμε τα

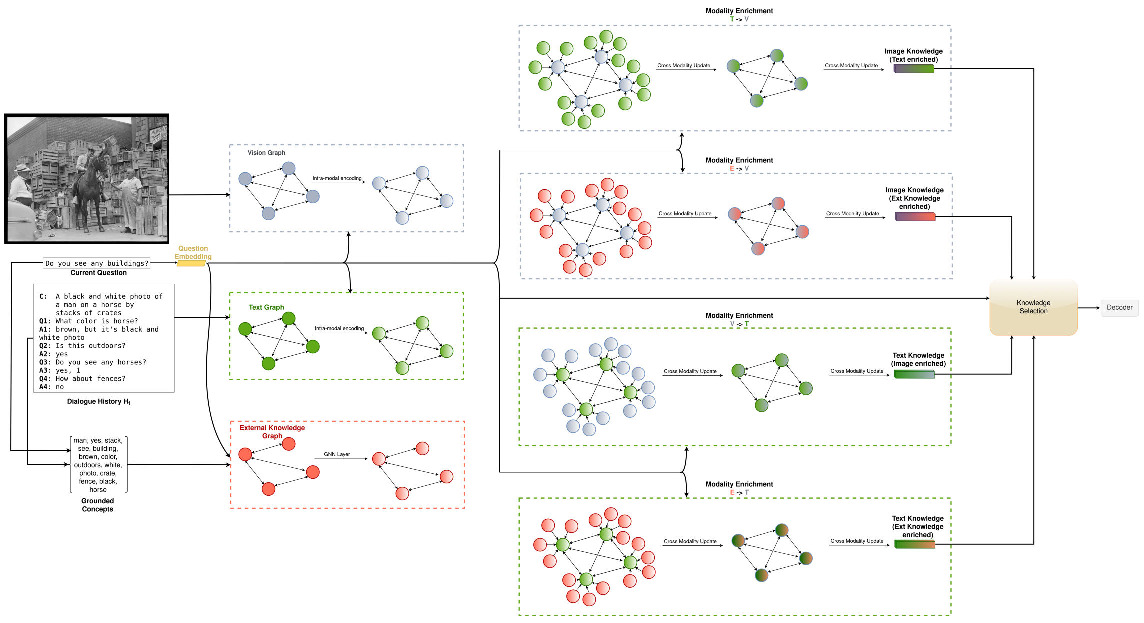

Do you see any buildings?
Current Question
Question Embedding
C: A black and white photo of a man on a horse by stacks of crates
Q1: What color is horse?
A1: brown, but it's black and white photo
Q2: Is this outdoors?
A2: yes
Q3: Do you see any horses?
A3: yes, 1
Q4: How about fences?
A4: no
Dialogue History $H_t$
man, yes, stack, see, building, brown, color, outdoors, white, photo, crate, fence, black, horse
Grounded Concepts
Vision Graph
Intra-modal encoding
Text Graph
Intra-modal encoding
External Knowledge Graph
GNN Layer
Modality Enrichment T -> V
Cross Modality Update
Image Knowledge (Text enriched)
Modality Enrichment E -> V
Image Knowledge (Ext Knowledge enriched)
Modality Enrichment V -> T
Text Knowledge (Image enriched)
Modality Enrichment E -> T
Text Knowledge (Ext Knowledge enriched)
Knowledge Selection
Decoder

**Figure 2.** *Η αρχιτεκτονική του προτεινόμενου μοντέλου*

παρακάτω πειράματα:

- **GCN [14]** As Graph Convolution Networks do not take into account multiple types of relations, this layer considers only one type. The relationship *related_ to* was selected for this purpose.

  Επιλέγοντας μόνο έναν τύπο ακμής από τον αρχικό γράφο, οι ανανεωμένες αναπαραστάσεις κάθε κόμβου προκύπτουν ένα κανονικοποιημένο σταθμισμένο άθροισμα των αναπαραστάσεων των κόμβων. Προκειμένου να ενσωματώσουμε την ερώτηση διακρίνουμε τις δύο περιπτώσεις:

  – Απλή συνένωση με την ερώτηση:

$$h_v^k = \sigma(W_k \sum_{u(v)} \frac{[h_v^{k-1}, Q_t]}{deg(v)}) \tag{5}$$

  – Τροποποίηση των γειτονικών κόμβων πριν το άθροισμα, υπολογίζοντας συντελεστές υπεύθυνους να δώσουν έμφαση στη σχετική με την ερώτηση πληροφορία

$$h_{vq}^{k-1} = softmax(W_2(Q_t \circ W_1 h_v^{k-1})) * h_v^{k-1} \tag{6}$$

$$h_v^k = \sigma(W_k \sum_{u(v)} \frac{h_{vq}^{k-1}}{deg(v)}) \tag{7}$$

- **R-GCN[27]** Ο πιο διαδεδομένος τρόπος για κωδικοποίηση γράφων με πολλάπλά είδη ακμών:

$$h_i^{(l+1)} = \sigma(\sum_{r \epsilon R} \sum_{j \epsilon N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0(l) h_i^{(l)} \tag{8}$$

  Για να αντιμετωπίσουμε τον κίνδυνο που ελλοχεύει το μοντέλο να μην εκπαιδευτεί σωστά λόγω των πολύ διαφορετικών συχνοτήτων εμφάνισης των διαφόρων τύπων ακμών, όπως φαίνεται στην υποενότητα 7.2.2, χρησιμοποιούμε όπως πρότεινε ο [27], κοινά βάρη για το κάθε είδος ακμής, τα οποία προκύπτουν όπως φαίνεται παρακάτω:

$$W_r^{(l)} = \sum_{b=1}^{B} a_{rb}^{(l)} V_b^{(l)} \tag{9}$$

Με αυτόν τον τρόπο επιτυγχάνουμε τόσο μείωση των παραμέτρων όσο και κοινή ανανέωση των παραμέτρων για όλα τα είδη ακμών.

- **GAT[4]** Στο συγκεκριμένο πειραμα εφαρμόζουμε μηχανισμό προσοχής:

$$a_{ij} = \frac{exp(LeakyReLU(a^T[Wh_i \parallel Wh_j]))}{\sum_{k \epsilon N_i} exp(LeakyReLU(a^T[Wh_i \parallel Wh_k]))} \tag{10}$$

$$h_i' = \sigma(\sum_{j \epsilon N_i} a_{ij} W h_j) \tag{11}$$

όπου $a_{ij}$ είναι ο συντελεστής προσοχής του κόμβου $j$ στον κόμβο $i$ και $h_i$ είναι οι τελικές αναπαραστάσεις του κόμβου $v_i$.

Όπως πρότεινε και ο [4] χρησιμοποιούμε πολλάπλη μέθοδο προσοχής. Συγκεκριμένα, εφαρμόζουμε K (K=8) ανεξάρτητους μηχανισμούς προσοχής με βάση τις παραπάνω εξισώσεις και εξάγουμε τις τελικές αναπαραστάσεις κάθε κόμβου υπολογίζοντας τον μέσο όρο από το αποτέλεσμα του κάθε μηχανισμού:

$$h_i' = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \epsilon N_i} a_{ij}^k W^k h_j) \tag{12}$$

Επιπρόσθετα, προκειμένου να χρησιμοποιήσουμε την τρέχουσα ερώτηση για να καθοδηγήσουμε την όλη διαδικασία, δοκιμάστηκε η συνένωση της τρέχουσας ερώτησης $Q_t$ με τις αναπαραστάσεις των κόμβων. Έτσι οι παραπάνω εξισώσεις καταλήγουν:

$$a_{ij} = \frac{exp(LeakyReLU(a^T[W[h_i \parallel Q_t] \parallel W[h_j \parallel Q_t]]))}{\sum_{k \epsilon N_i} exp(LeakyReLU(a^T[W[h_i \parallel Q_t] \parallel W[h_k \parallel Q_t]]))} \tag{13}$$

$$h_i' = \sigma(\sum_{j \epsilon N_i} a_{ij} W[h_j \parallel Q_t]) \tag{14}$$

$$h_i' = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \epsilon N_i} a_{ij}^k W^k [h_j \parallel Q_t]) \tag{15}$$

- **Μέθοδος προώθησης μηνύματος** Τέλος, πειραματιζόμαστε με έναν διαφορετικό τρόπο κωδικοποίησης εμπνευσμένο από τον [28]. Συγκεκριμένα, κατασκευάζουμε τις ακμές του γράφου ως τη συνένωση μεταξύ των δύο γειτονικών κόμβων:

$$e_{ij} = [v_i, v_j] \tag{16}$$

όπου το "$[\cdot, \cdot]$" δηλώνει τη συνένωση.

Στη συνέχεια ανανεώνουμε την αναπαράσταση των ακμών χρησιμοποιώντας την τρέχουσα ερώτηση $Q_t$, προκειμένου να κατασκευάσουμε τελικά τους συντελεστές $a$:

$$a_{ij} = softmax(W_e(W_1 Q_t \circ W_2 e_{ij})) \tag{17}$$

$$\hat{e}_{ij} = a_{ij} e_{ij} \tag{18}$$

Τελικά, εφαρμόζουμε ένα σταθμισμένο άθροισμα σε κάθε κόμβο:

$$\beta_{ij} = softmax(W_v(Q_t \circ W_3[u_j, \hat{e}_{ij}])) \tag{19}$$

$$\hat{v}_i = \sum_{j=1}^{N} \beta_{ij} u_j \tag{20}$$

Έχοντας κωδικοποιήσει την πληροφορία από κάθε πηγή ξεχωριστά στη συνέχεια εμπλουτίζουμε τη πληροφορία που προέρχεται από την Εικόνα και τον Διάλογο με όλες τις άλλες πηγές πληροφορίας.

Θεωρώντας ότι θέλουμε να εμπλουτίσουμε την τροπικότητα M1 που αναπαριστάται από τον γράφο $G^1 : (V^1, E^1)$ με την τροπικότητα M2, που αναπαριστάται από τον γράφο $G^2 : (V^2, E^2)$, ακολουθάμε την παρακάτω διαδικασία:

1. Κατασκευή του M2-to-M1 γράφου, όπου κάθε κεντρικός κόμβος $v_i^1$ συνδέεται με όλους τους κομβους του συνόλου $V^2 = v_i^2$ με ακμές $B_{ij}^1$, που προκύπτουν με τη συνένωση: $B_{ij}^1 = [v_i^1, v_j^2]$

2. Ενημέρωση των ακμών με την καθοδήγηση της τρέχουσας ερώτησης $Q_t$:

$$\gamma_{ij} = softmax(W_b^1(W_1^1 Q_t \circ W_5^1 B_{ij}^1)) \tag{21}$$

$$\hat{B}_{ij}^1 = \gamma_{ij} B_{ij}^1 \tag{22}$$

3. Ενημέρωση των κεντρικών κόμβων:

$$\delta_{ij}^1 = softmax(W_c^1(Q_t \circ W_6^1[v_j^2, \hat{B}_{ij}^1])) \tag{23}$$

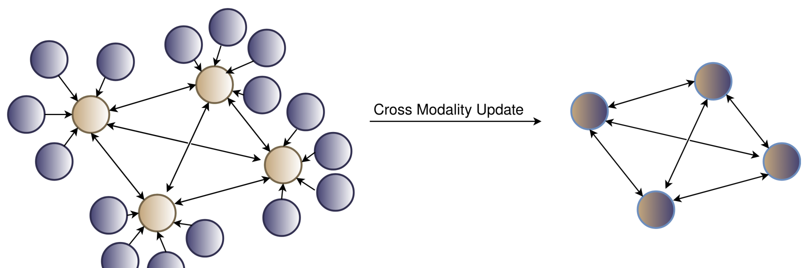


**Figure 3.** *Ενημέρωση των κεντρικών κόμβων $V^1$ (καφε) αθροίζοντας ολόυς τους γύρω κόμβους $V^2$ (μπλε).*

$$\hat{v}_i^1 = \sum_{j=1}^{N^2} \delta_{ij} v_j^2 \tag{24}$$

4. Εφαρμογή μηχανισμού πυλών μεταξύ των αρχικών αναπαραστάσεων των κεντρικών κόμβων και αυτών που μόλις υπολογίσαμε $\hat{v}_i^1$:

$$gate_l^{v^1} = \sigma(W_l^{v^1}[v_i^1, \hat{v}_i^1]) \tag{25}$$

$$\hat{v}_i^{1l} = W_7^1(gate_l^{v1} \circ [v_i^1, \hat{v}_i^1]) \tag{26}$$

5. Υπολογισμός αναπαράστασης του γράφου που προέκυψε από την αρχική κωδικοποίηση:

$$\eta_i^{v^1} = softmax(W_e^{v^1}(Q_t \circ (W_8^1 \hat{v}_i^1))) \tag{27}$$

$$\hat{V}_o^1 = \sum_{i=1}^{N^1} \eta_i^{v^1} \hat{u}_i^1 \tag{28}$$

6. Υπολογισμός αναπαράστασης του γράφου που προέκυψε από τον εμπλουτισμό της τροπικότητας M1 από την τροπικότητα M2:

$$\mu_i^{v^1} = softmax(W_\alpha^{v^1}(Q_t \circ (W_9 \hat{v}_i^1 l))) \tag{29}$$

$$\hat{V}_c^1 = \sum_{i=1}^{N} \mu_i^{v^1} \hat{u}_i^{1l} \tag{30}$$

7. Τέλος, υπολογισμός της συνολικής γνώσης για την τροπικότητα M1 εμπλουτισμένη από την M2 εφαρμόζοντας μηχανισμό πύλης στις δύο αναπαραστάσεις γράφου που εξάγαμε προηγουμένως:

$$gate_g^{v^1} = \sigma(W_l^g[\hat{V}_o^1, \hat{V}_c^1]) \tag{31}$$

$$\hat{V}^1 = W_{10}(gate_g^{v^1} \circ [\hat{V}_o^1, \hat{V}_c^1]) \tag{32}$$

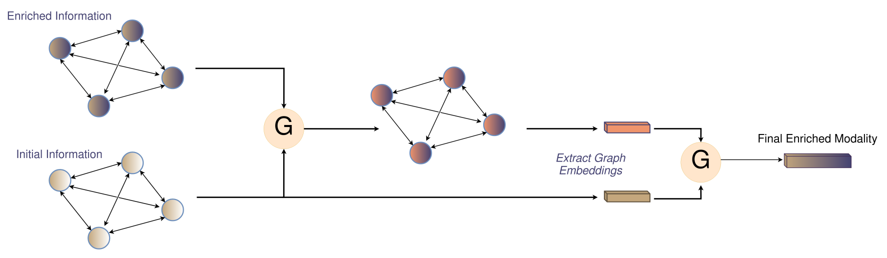


**Figure 4.** *Εφαρμογή μηχανισμού πύλης αρχικά σε επίπεδο κόμβων και στη συνέχεια σε επίπεδο γράφων.*

Χρησιμοποιούμε αυτήν τη διαδικασία προκειμένου να τη χρησιμοποιήσουμε στα παρακάτω ζευγάρια τροπικοτήτων:

1. Εμπλουτισμός_τροπικότητας(Εικόνα, Διάλογος), για τον εμπλουτισμό της τροπικότητας της Εικόνας με πληροφορία από την τροπικότητα του Διαλόγου.

2. Εμπλουτισμός_τροπικότητας(Εικόνα, Εξωτερική Γνώση), για τον εμπλουτισμό της τροπικότητας της Εικόνας με πληροφορία από την τροπικότητα της Εξωτερικής Γνώσης

3. Εμπλουτισμός_τροπικότητας(Διάλογος, Εικόνα), για τον εμπλουτισμό της τροπικότητας του Διαλόγου με πληροφορία από την τροπικότητα της Εικόνας

4. Εμπλουτισμός_τροπικότητας(Διάλογος, Εξωτερική Γνώση), για τον εμπλουτισμό της τροπικότητας του Διαλόγου με πληροφορία από την τροπικότητα της Εξωτερικής Γνώσης

Η παραπάνω διαδικασία οδηγεί σε τέσσερεις διαφορετικές γνώσεις, οι οποίες θα αποτελέσουν την είσοδο για το τελευταίο μέρος του κωδικοποιητή του μοντέλου μας.

Έχοντας δημιουργήσει 2 είδη γνώσεων για τις τροπικότητες της Εικόνας ($\hat{I}^T$ ,$\hat{I}^E$) και του Διαλόγου ($\hat{T}^I$, $\hat{T}^E$), υπολογίζουμε τη συνολική γνώση για καθεμία από τις δύο τροπικότητες και τελικά τις συνδυάζουμε μαζί και με την τρέχουσα ερώτηση για να εξάγουμε την τελική αναπαράσταση $\hat{K}$, όπως φαίνεται στις παρακάτω εξισώσεις:

$$gate_i = \sigma(W_r[\hat{I}^T, \hat{I}^E]) \tag{33}$$

$$\hat{K}_I = W_{11}(gate_r \circ [\hat{I}^T, \hat{I}^E]) \tag{34}$$

$$gate_T = \sigma(W_r[\hat{T}^I, \hat{T}^E]) \tag{35}$$

$$\hat{K}_T = W_{12}(gate_r \circ [\hat{T}^I, \hat{T}^E]) \tag{36}$$

$$gate_r = \sigma(W_r[Q_t, \hat{K}_I, \hat{K}_T]) \tag{37}$$

$$\hat{K} = W_{13}(gate_r \circ [Q_t, \hat{K}_I, \hat{K}_T]) \tag{38}$$

Τα παραπάνω απεικονίζονται συνοπτικά στην εικόνα 5.

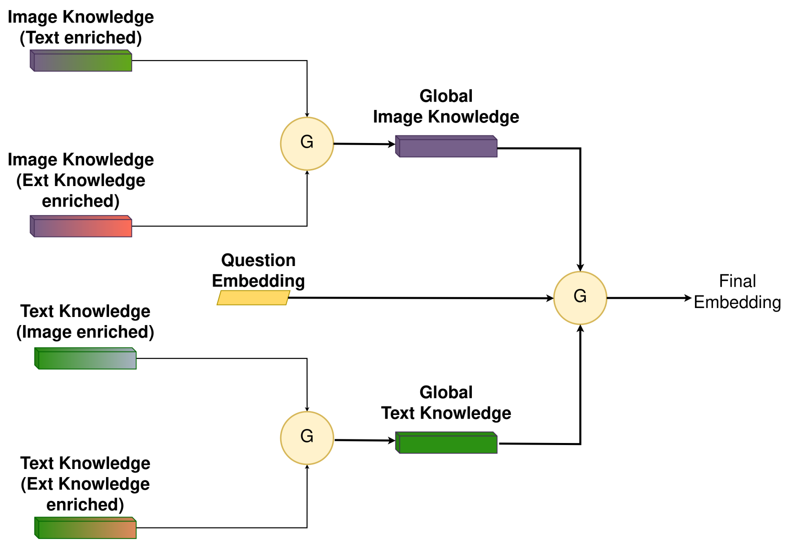


Image Knowledge (Text enriched)
Image Knowledge (Ext Knowledge enriched)
G
Global Image Knowledge
Question Embedding
Text Knowledge (Image enriched)
Text Knowledge (Ext Knowledge enriched)
G
Global Text Knowledge
G
Final Embedding

**Figure 5.** *Εφαρμογή μηχανισμών πυλών για τον υπολογισμό των συνολικών γνώσεων για την κάθε τροπικότητα και στη συνέχεια για τον συνδυασμό αυτων.*

## 0.6 Πειράματα

Δοκιμάζουμε τα πειράματά μας και με τους δύο αποκωδικοποιητές που προτάθηκαν από τον [11]

### 0.6.1 Διακρίνων αποκωδικοποιητής

Ο Διακρίνων αποκωδικοποιητής ταξινομεί όλες τις πιθανές απαντήσεις από το σύνολο των 100 υποψήφιων απαντήσεων $A$. Για το σκοπό αυτόν υπολογίζει για κάθε πιθανή απάντηση το εσωτερικό γινόμενο μεταξύ την κωδικοποιημένης πληροφορίας και της κωδικοποιημένης με ένα αναδρομικό Νευρωνικό Δίκτυο, το LSTM, απάντησης. Τα γινόμενα στη συνέχεια διοχετεύονται σε μία softmax συνάρτηση προκειμένου να υπολογιστεί η πιθανότητα κάθε απάντησης. Κατά τη διάρκεια της εκπαίδευσης του μοντέλου μεγιστοποιούμε την πιθανότητα της σωστής απάντησης, ενώ κατά την εκτίμηση οι επιλογές ταξινομούνται απλώς με βάση τις πιθανότητες που προέκυψαν.

Αποτελέσματα στο VisDial v1.0 dataset:

| Αποτελέσματα στο test split | | | | | | |
|---|---|---|---|---|---|---|
| | MRR ↑ | R@1 ↑ | R@5 ↑ | R@10 ↑ | Mean ↓ | NDCG ↑ |
| KBGN | 64.13 | 50.47 | 80.70 | 90.16 | 4.08 | 57.60 |
| KBGN-Impl | 62.82 | 48.95 | 80.33 | 89.23 | 4.28 | 56.62 |
| KBGN-Numb | 62.59 | 48.42 | 80.23 | 89.16 | 4.28 | 56.19 |
| KBGN-Ext-GCN-CON-Q | 63.25 | 49.43 | 80.05 | 89.63 | 4.21 | 55.79 |
| KBGN-Ext-GCN-S-Q | 62.78 | 48.95 | 79.48 | 89.0 | 4.32 | 55.57 |
| KBGN-Ext-GAT | 55.69 | 42.7 | 70.18 | 79.73 | 7.87 | 51.87 |
| KBGN-Ext-GAT-Q | 56.21 | 43.04 | 70.85 | 80.02 | 7.79 | 51.98 |
| KBGN-Ext-RGCN | 62.873 | 49.2 | 79.35 | 89.15 | 4.33 | 55.78 |
| KBGN-MessagePassing | 62.71 | 48.93 | 80.32 | 89.21 | 4.11 | 56.58 |

**Table 1.** *Αποτελέσματα στο test split χρησιμοποιώντας τον Διακρίνων αποκωδικοποιητή.*

| Results on validation split | | | | | | |
|---|---|---|---|---|---|---|
| | MRR ↑ | R@1 ↑ | R@5 ↑ | R@10 ↑ | Mean ↓ | NDCG ↑ |
| KBGN | 64.86 | 51.37 | 81.71 | 90.54 | 4.00 | 59.08 |
| KBGN-Impl* | 63.84 | 50.04 | 80.87 | 90.02 | 4.104 | 57.056 |
| KBGN-Numb | 63.61 | 49.69 | 80.68 | 89.93 | 4.11 | 56.64 |
| KBGN-Ext-GCN-CON-Q | 64.04 | 50.38 | 81.16 | 89.92 | 4.15 | 56.58 |
| KBGN-Ext-GCN-S-Q | 63.60 | 49.73 | 80.66 | 89.97 | 41.15 | 55.74 |
| KBGN-Ext-GAT | 69.41 | 56.07 | 86.3 | 93.84 | 3.16 | 53.42 |
| KBGN-Ext-GAT-Q | 70.68 | 57.31 | 87.23 | 94.29 | 3.10 | 54.57 |
| KBGN-Ext-RGCN | 63.7 | 49.93 | 80.59 | 89.95 | 4.12 | 55.63 |
| KBGN-MessagePassing | 63.91 | 50.09 | 80.90 | 89.95 | 4.1 | 56.52 |

**Table 2.** *Αποτελέσματα στο val split χρησιμοποιώντας τον Διακρίνων αποκωδικοποιητής.*

Στους παραπάνω πίνακες συμβολίζουμε με *KBGN-Ext-GCN-CON-Q* και *KBGN-Ext-GCN-S-Q* τα μοντέλα που περιγράφονται από τις σχέσεις 0.5. Το πρώτο εφαρμόζει συνένωση μεταξύ των αναπαραστάσεων της ερώτησης και κόμβων, ενώ το δεύτερο χρησιμοποιεί την ερώτηση για τον υπολογισμό συντελεστών για κάθε κόμβο. Τα μοντέλα *KBGN-Ext-GAT* και *KBGN-Ext-GAT-Q* σε αυτά που περιγράφονται από το μοντέλο 5.5.2, με και χωρίς να λαμβάνουμε υπόψην την ερώτηση αντίστοιχα. Με *KBGN-Ext-RGCN* συμβολίζουμε το μοντέλο 5.5.2, ενώ με *KBGN-MessagePassing* το μοντέλο 5.5.2. Τέλος, το μοντέλο *KBGN-Numb* αναφέρεται ακριβώς στο μοντέλο που χρησιμοποιήσαμε ως αφετηρία 5.4, αλλά χρησιμοποιώντας τα Numberbatch embeddings αντί για τη συνένωση των GloVe και ELMo.

### 0.6.2 Γεννητικός Αποκωδικοποιητής

Ο αποκωδικοποιητής αυτό υπολογίζει την πιθανότητα κάθε λέξης του λεξιλογίου σε κάθε βήμα του. Το διάνυσμα της κωδικοποιημένης πληροφορίας απο τον κωδικοποιητή αποτελεί την αρχική του κατάσταση ενός αναδρομικού Νευρωνικού Δικτύου, LSTM. Κατά τη διάρκεια της εκπαίδευσης μεγιστοποιούμε την πιθανότητα της ακολουθίας λέξεων της σωστής απάντησης. Κατά την εκτίμηση, ταξινομούμε τις υποψήφιες απαντήσεις, χρησιμοποιώντας τις πιθανότητες που εξάγει το μοντέλο. Αξίζει να σημειωθεί αφού ότι αυτός ο αποκωδικοποιητής δε ταξινομεί πιθανές απαντήσεις κατά την εκπαίδευση, το μοντέλο που τον χρησιμοποιεί δεν μπορεί να εκμεταλλευτεί τυχόν bias που δύναται να υπάρχει στις πιθανές ερωτήσεις της κάθε ερώτησης και για αυτό τον λόγο τείνει να παρουσιάζει χειρότερη απόδοση. Παρ' όλα αυτά τέτοιου είδους αποκωδικοποιητές είναι πιο χρήσιμοι, καθώς δύναται να χρησιμοποιηθούν σε ρεαλιστικές εφαρμογές, όπου δεν θα υπάρχει πάντα ένα σύνολο υποψήφιων απαντήσεων.

| Results on validation split | | | | | | |
|---|---|---|---|---|---|---|
| | MRR ↑ | R@1 ↑ | R@5 ↑ | R@10 ↑ | Mean ↓ | NDCG ↑ |
| KBGN | 50.05 | 40.40 | 60.11 | 66.82 | 17.54 | 60.42 |
| KBGN-Impl | 48.09 | 39.04 | 56.97 | 62.58 | 20.82 | 56.94 |
| KBGN-Numb | 47.88 | 38.74 | 56.62 | 62.03 | 20.95 | 56.42 |
| KBGN-Ext-GCN-CON-Q | 48.23 | 39.05 | 57.39 | 63.35 | 20.72 | 57.31 |
| KBGN-Ext-GCN-S-Q | 48.01 | 38.98 | 56.91 | 62.53 | 20.96 | 56.84 |
| KBGN-Ext-GAT | 51.01 | 41.73 | 60.75 | 66.58 | 19.18 | 55.46 |
| KBGN-Ext-GAT-Q | 51.36 | 42.03 | 61.38 | 66.97 | 19.06 | 55.58 |
| KBGN-Ext-RGCN | 48.15 | 39.03 | 57.01 | 62.89 | 20.78 | 56.27 |
| KBGN-MessagePassing | 48.12 | 38.93 | 56.95 | 63.19 | 21.78 | 57.25 |

**Table 3.** *Αποτελέσματα στο val split χρησιμοποιώντας τον Γεννητικό αποκωδικοποιητή.*

## 0.7 Σχολιασμός αποτελεσμάτων

Σαν ένα πρώτο γενικό σχόλιο πάνω στα αποτελέσματα, παρατηρούμε ότι μοντέλα που επιτυγχάνουν καλά αποτελέσματα χρησιμοποιώντας τον έναν αποκωδικοποιητή, παρουσιάζουν καλές επιδόσεις και στον άλλο. Οι χειρότερη απόδοση του γεννητικού αποκωδικοποιητή είναι όπως αναφέραμε στην περιγραφή του αναμενόμενη. Παρόλο που και οι δύο αποκωδικοποιητές είχαν προταθεί από τους δημιουργούς του Οπτικού Διαλόγου [11], η πλειοψηφία των ερευνητών ανακοινών αποτελέσματα μόνο για τον διακρίνων αποκωδικοποιητή.

Βασιζόμενοι στα νούμερα των πειραμάτων στους πίνακες 5.4 και 5.5 παρατηρούμε ότι τα αποτελέσματά μας είναι συγκρίσιμα με τα αποτελέσματα της δικής μας υλοποίησης του αρχικού μοντέλου, ωστόσο μόνο κάποια από αυτά καταφέρνουν να το ξεπεράσουν. Συγκεκριμένα, το μοντέλο KBGN-Ext-GCN-CON-Q, παρόλο που είναι το πιο απλό από όσα δοκιμάστηκαν, επιτυγχάνει τα καλύτερα αποτελέσματα και καταφέρνει να ξεπεράσει το αρχικό μας μοντέλο κατά 0.43% στο test-split. Επίσης, το μοντέλο KBGN-Ext-RGCN επιτυγχάνει και αυτό λίγο καλύτερα αποτελέσματα από το αρχικό. Μία πιθανή εξήγηση για το ότι δεν καταφέρνει καλύτερα αποτελέσματα είναι η μεγάλη διαφορά στις συχνότητες εμφάνισης των διαφόρων τύπων ακμών στον γράφο Εξωτερικής Γνώσης. Όπως φαίνεται στα γραφήματα 7.3, 7.4 και 7.5 ο τύπος ακμής *related_to* είναι με διαφορά κυρίαρχος σε αυτόν το γράφο. Παρόλο που στο συγκεκριμένο πείραμα αποκλεισαμε τα εξαιρετικά σπάνια είδη ακμών, η ανισορροπία παραμένει πολύ έντονη. Το μοντέλο KBGN-MessagePassing αν και ακολουθεί μία προσέγγιση παρόμοια με αυτήν στη κωδικοποίηση των γράφων των άλλων δύο τροπικοτήτων, Εικόνες και Κειμένου, δεν καταφέρνει να επιτύχει καλά αποτελέσματα. Μία πιθανή εξήγηση είναι είναι ότι αυτοί οι γράφοι έχουν εντελώς διαφορετική δομή.

## 0.8 Συμπεράσματα

Στην παρούσα εργασία μελετάμε το πρόβλημα του Οπτικού Διαλόγου. Η προσέγγισή μας χρησιμοποιεί Νευρωνικά Δίκτυα Γράφων, έναν τύπο δικτύων που έχει αποδειχθεί ιδιαίτερα αποτελεσματικός σε μεγάλο φάσμα προβλημάτων. Πειραματιζόμαστε με την εισαγωγή εξωτερικής γνώσης στο μοντέλο και με τον τρόπο που θα την κωδικοποιήσουμε προκειμένου να βελτιώσει τα αποτελέσματα. Για το σκοπό αυτό, κατασκευάζουμε τρείς γράφους, έναν για κάθε τροπικότητα και πραγματοποιούμε τρία στάδια κωδικοποίησης και σύμμειξης αυτών.

Αξιολογούμε τα μοντέλα μας χρησιμοποιώντας δύο είδη αποκωδικοποιητών, τον διακρίνων και τον γεννητικό. Παρατηρούμε με βάση τα αποτελέσματα στους πινακες 5.4, 5.5 και 5.6 ότι μοντέλα που επιτυγχάνουν καλά αποτελέσματα χρησιμοποιώντας τον έναν αποκωδικοποιητή, έχουν καλές αποδόσεις χρησιμοποιώντας και τον άλλο. Η συνέπεια αυτή μεταξύ των καλύτερων και χειρότερων μοντέλων τονίζει την επίδραση των διαφορετικών κωδικοποιητών στη συνολική απόδοση του μοντέλου.

Με βάση τα αποτελέσματα, μπορούμε να εξάγουμε συμπεράσματα σχετικά με τη σημασία της ύπαρξης του γράφου εξωτερικής γνώσης αλλά και των διαφορετικών τρόπων κωδικοποίησής της. Συγκεκριμένα, καταφέρνουμε να ξεπεράσουμε τα αποτελέσματα του υλοποιημένου από εμάς αρχικού μοντέλου, χρησιμοποιώντας τις μεθόδους *KBGN-Ext-GCN-CON-Q* και *KBGN-Ext-RGCN*. Η πρώτη λαμβάνει υπόψην μόνο τον τύπο ακμής *related_to*, ενώ η δεύτερη πολλαπλούς τύπους ακμών. Οι μέθοδοι που εφαρμόζουν Αυτο-Προσοχή πάνω στους κόμβους, *KBGN-Ext-GAT* και *KBGN-Ext-GAT-Q*, του γράφου εξωτερικής γνώσης επιτυγχάνουν καλά αποτελέσματα μόνο στο validation split, γεγονός που μπορεί να οφείλεται σε λάθος ρύθμιση των υπερπαραμέτρων. Η μέθοδος *KBGN-Ext-MessagePassing* αν και η πιο σύνθετη δεν επιτυγχάνει καλά αποτελέσματα και αυξάνει σημαντικά τον αριθμό των παραμέτρων. Τέλος, η μέθοδος *KBGN-Numb*, που είναι το αρχικό μοντέλο αλλά χρησιμοποιώντας αναπαραστάσεις λέξεων που προκύπτουν από τον γράφο γνώσης που έχουμε χρησιμοποιήσει επιτυγχάνει τα χειρότερα αποτελέσματα. Αυτό επιδεικνύει τη σημασία ύπαρξης του γράφου εξωτερικής γνώσης, ο οποίος εισάγει πολλή περισσότερη πληροφορία.

Συμπερασματικά, μπορούμε να εξάγουμε το συμπέρασμα ότι η εισαγωγή εξωτερικής γνώσης στο μοντέλο οδηγεί σε καλύτερα αποτελέσματα. Οι μέθοδοι που κατάφεραν καλύτερες επιδόσεις ήταν οι *KBGN-Ext-GCN-CON-Q* και *KBGN-Ext-RGCN*. Τα αποτελέσματα αυτά είναι λογικά. Αναφορικά με την πρώτη, αντίστοιχη μέθοδος κωδικοποίησης εξωτερικής γνώσης έχει χρησιμοποιηθεί σε παρόμοιο πρόβλημα με τον οπτικό διάλογο, όπως στο [29]. Επιπρόσθετα, η δεύτερη μέθοδος που πέτυχε λίγο καλύτερα αποτελέσματα αποτελεί μία από τις πιο δημοφιλείς τεχνικές για την κωδικοποίηση γράφων με πολλαπλά είδη ακμών. Τέλος, πρέπει να τονίσουμε τη σημασία της προεπεξεργασίας του γράφου γνώσης ConceptNet, ο οποίος χρησιμοποιήθηκε για τη δημιουργία του External Knowledge Graph για κάθε διάλογο. Οι σχεδιαστικές επιλογές του αλγορίθμου και η επιθυμητη λειτουργία του είναι καθοριστικές για την επίτευξη ενός χρονικά αποδοτικού αλγορίθμου, αλλά και για την επιλογή χρήσιμης πληροφορίας από τον γράφο γνώσης.

Chapter 1

# Introduction

## 1.1 What is Artificial Intelligence?

Artificial Intelligence is the ability of a digital computer to perform tasks commonly associated with intelligent beings. These tasks vary from visual perception, speech recognition and translation between languages to decision-making and taking actions in a given environment. Hence, the term Artificial Intelligence can be considered as a more general field, that contains a lot of sub-fields that emerged through the years, such as Machine Learning, Deep Learning and Reinforcement Learning.

One of the first approaches to artificial intelligence is the **Knowledge-Based approach**, which differs a lot to some of the newer developments in AI. Specifically, a knowledge-based system is comprised of a knowledge base, contains a collection of information in a given field, and an inference engine, which deduces insights from the information stored in the knowledge base. These approaches, however, suffer from the complexity and difficulty to sufficiently describe all the knowledge regarding the given task.

These problems were tackled by another field of Artificial Intelligence, **Machine Learning**. In this approach, the system tries to recognize patterns from the raw data and make decisions with minimal human intervention. Various types of models have been used and researched for machine learning systems, some of which are:

- **Artificial neural networks**, which are inspired by the biological neural networks and "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules,
- **Decision trees**, where learning uses a decision tree as a predictive model to go from observations about an item (branches) to conclusions about the item's target value (leaves)
- **Support-vector machines** (SVMs), where given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other.
- **Bayesian network** a probabilistic graphical model that represents a set of random variables and their conditional independence with a directed acyclic graph (DAG).

The performance of these machine learning algorithms depends directly on the representation and the amount of the given data. One of the most important drawbacks of these algorithms is the difficulty to extract high-level features from the input raw data.

**Deep Learning** comes to address this crucial problem in representation learning by introducing representations that are expressed in terms of other simpler representations. This approach manages to build complex concepts out of simpler concepts, but demands enormous data.

## 1.2 Motivation

Dialog has always been the most effective way for humans to exchange information. Due to this significance, it is an important research goal to develop artificial intelligence based agents capable of conducting human-computer conversation. However, during a human conversation, subtle details and nuances are often very important. This importance of subtle details and nuances makes the development of agents for visual dialog a challenging endeavor. Recent efforts to facilitate human-computer conversation about images, focus on image captioning, visual question answering, visual question generation and, lastly, also visual dialog. These tasks have been using a set of very general datasets, that attempt to imitate the human conversation on images. To this end, Das et al. [11] has collected, curated and provided to the general public an impressive dataset, which enables to design virtual assistants that can converse.

Most tasks related to Natural Language Processing have been approached with classical methods, studying the fusion of the current question, dialogue history and image via using attention mechanism [30]. In this paper we examine the use of Graph Neural Networks, which the last years have achieved state-of-the-art results in many tasks and have more recently been introduced to Natural Language Processing. In addition, a realistic dialogue between two humans would necessarily contain some common knowledge from the external world. Towards this direction, we experiment on equipping our model with external knowledge, from a Knowledge Graph.

## 1.3 Research Contributions

In this study, we examine a number of multimodal fusion methods using Graph Neural Networks. Visual dialog is an on-going conversation about the image, and the relations among visual objects are dynamically altered with conversational contexts. As a result, retrieving information from both vision (image) and text modality (dialogue history) is required. [11]. Between these two modalities, however, there exists a heterogeneous semantic gap of implicit referring relations cross modalities. The modeling of which can be crucial to our task and not only. Fusing therefore the two modalities in a proper way can unlock way better results.

Moreover, equipping the model with external knowledge, which we will be using as a third modality will lead not only to a numerous of possible fusion methods, but also to conclusions on whether the knowledge of the external world is crucial in order to imitate a human to human conversation on an image.

Finally, the use of Graph Neural Networks, additionally, provides a whole new approach on the ways to implement the encoding and fusion steps. GNNs are attracting more and more research attention on a vast variety of machine learning problems. Employing them on a task that does not necessarily demand a graph approach, demonstrates the how well these networks can adapt and model various types of problems.

## 1.4 Thesis Outline

The structure of the rest of the thesis is as follows:

Chapter 2: **Machine Learning**, provides a theoretical background knowledge that will be needed for the following chapters. First, we give an overview of technical information, that is relevant to the basics concepts of Machine Learning. Then we provide a swift description of the different types of learning and a detailed introduction to more complex Deep Learning methods. Finally we dig in the concepts that are primarily used in this thesis.

Chapter 3: **Graph Neural Networks**, provides a detailed introduction into Graph Neural Networks, which are widely used in this thesis. After analyzing the motivation behind using these neural networks and their advantages, we describe the original Graph Neural Network as it was initially proposed and enumerate its limitations on a set of tasks. Then we move on presenting a few general Graph Neural Networks frameworks and then describing the most popular models that were introduced through the years. Finally, a brief presentation of Graph Knowledge Bases is provided.

Chapter 4: **Introduction to NLP and Visual Dialog**, provides a detailed presentation of our task and an overview of the related fields; natural language processing, computer vision and multimodal fusion as well as the background information needed to comprehend this thesis.

Chapter 5: **Proposed Model**, firstly provides a detailed dataset description and then a brief survey on the related to the task work. Subsequently, a detailed presentation of our approach is analyzed. We experiment with different methods of utilizing the external knowledge and fusing it with the image and text modality. We then demonstrate the impact that the external knowledge has on this specific task.

Chapter 6: **Conclusions**, contains our conclusion, summarizing our findings and providing an outlook into the future work.

Chapter 2

# Machine Learning

## 2.1 Introduction

Because of the huge and rapid steps that have been on the field through the last years, the terms Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) are usually confused. Artificial Intelligence can be defined as the ability of a digital computer to perform tasks commonly associated with intelligent beings. These tasks vary from visual perception, speech recognition and translation between languages to decision-making and taking actions in a given environment. Hence, the term Artificial Intelligence can be considered as a more general field, sub-fields of which are AI, ML, and DL, and the distinction is based on the technique in which the computer systems can mimic human intelligence.

Machine Learning is the area of research that studies algorithms that are able to improve on a given task without being explicitly programmed. Machine learning algorithms leverage huge amounts of data and extract patterns in order to "learn" how to perform on a given problem. This approach goes beyond traditional AI methods, which used to extract new knowledge and reason about statements, through logical inference rules and suffered from the difficulty of formally describing all the knowledge based on the given task. Machine Learning algorithms have been proven effective against a vast variety of problems, where an explicit deterministic algorithm was to difficult to be developed. Tasks such as speech recognition and computer vision have seen unprecedented progress through the last years, while solutions to problems from other sciences, such as medicine have also been boosted using machine learning techniques.

In general most Machine Learning algorithms attempt to extract a mapping representation from the representation space to the output space, while others intend to learn the representation itself. The field the last ones belong to is called representation learning. Nevertheless, extracting high-level features from data can be very challenging depending on the dataset and also on various factors such as noise.

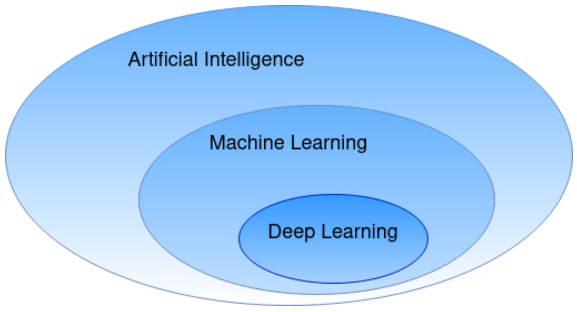


**Figure 2.1.** *Artificial Intelligence, Machine Learning and Deep Learning Domains*

## 2.2 Learning Categories

The most modern definition of (machine) learning was provided by Tom Mitchell. According to this definition: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”. In other words, when providing input data to our model, we expect after the training procedure, that its capability of completing a given task will be increased. There exists various different types of learning, which can be grouped in four main categories:

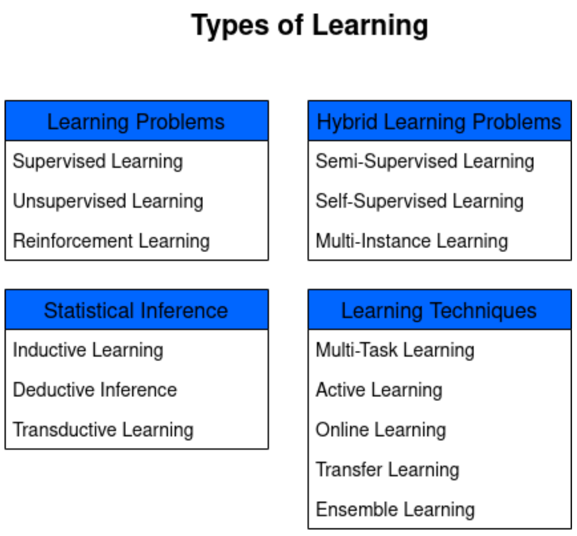


**Figure 2.2.** *Types of Learning*

The most common types of learning are being presented in the following subsections.

### 2.2.1 Supervised Learning

Supervised Learning describes a class of problems, where the model aims to learn a mapping between input examples and the target variable. To formally model the above, the goal of this type of learning is to learn a mapping function that will map the input variables $X$ of the model to the output variable $Y$:

$$Y = f(X) \tag{2.1}$$

In supervised learning the training data are labeled. Specifically, the training procedure consists of supplying the model both with $X$ and their corresponding $Y$. Subsequently, the model makes predictions on test sets where only the inputs are provided and the outputs from the model are compared to the groundtruth target variables and used to estimate the skill of the model. During inference, we expect that the mapping function can successfully predict the output for every given $X$ provided from the same distribution as the training sample.

There are two main types of supervised learning problems, distinguished by the nature of the output $Y$:

1. **Classification**: Supervised learning problem that involves predicting a class label, the label of the input data $X$.

2. **Regression**: Supervised learning problem that involves predicting a numerical value.

### 2.2.2 Unsupervised Learning

As opposed to supervised machine learning problems, where the data consists of input vectors and their labels, there a lot of problems that don't provide any target value. Unsupervised learning is a type of algorithm that intends to extract patterns from these kinds of datasets. Based on these patterns the model is expected to assign the samples to a class or a value.

One important subclass of unsupervised tasks is the problem clustering. Clustering is the process of categorizing a population or set of data points into a number of groups, so that similar data points will belong to the same group and will be different from data points of other groups.

Generative modeling is another intriguing category of unsupervised tasks. Models that emulate the process of generating the training data are known generative models. A good generative model will produce new data that is similar to the training data in some way. Because the mechanism that generates the data is not clearly observable, this sort of learning is considered unsupervised.

### 2.2.3 Semi-Supervised Learning

Semi-Supervised Learning is a machine learning technique that involves training with a small amount of labeled data and a large number of unlabeled data. This method is a combination of Supervised and Unsupervised learning and belongs to the category of the Hybrid Learning Problems. It is used for datasets that it is impossible to label every sample, so the model approaches the problem by firstly using the existing labeled samples to generate pseudo-labels for the unlabeled samples and then utilizes the whole dataset combined.

## 2.3 Supervised Machine Learning Algorithms

In this work, as it will be discussed in detail in Section 5.9, we employ Supervised learning to train our model to cope effectively with the given task. Consequently, the following sections will provide background primarily on supervised learning methods. Below is a more formal definition of Supervised learning:

**Definition**: Given a dataset of N training examples $D = \{(x_n, y_n), n = 1, \cdots, N\}$, the task is to learn a function $f : X \to Y$ mapping the input X to the output space Y . How well the function f fits the training data, i.e. how accurately it maps X to Y , is quantified by a loss function $L : Y \times Y \to \mathbf{R}_{\geq 0}$ . For instance, given a training example $(x_i, y_i)$ the loss of predicting the value $\hat{y}_i = f(\mathbf{x}_i)$ is computed by $L(\mathbf{\hat{y}_i}, \mathbf{y_i})$.

### 2.3.1 Activation Function

In artificial neural networks, the activation function refers to the equation that determines the output of each neuron given its input. In other words, it defines how the weighted sum of the input is transformed into an output. Each neuron in a neural network can be equipped with a different activation function, the selection of which can be very crucial. There has been used a wide variety of different activation functions through the years, that can be grouped into different categories. The simplest activation function can be defined as a binary function that "activates" the neuron, based on its input and outputs "0" or "1". Activation functions can be classified either as linear or as non-linear. Each category and the most widely used activation functions are presented below:

**Linear Activation Function**: A linear function is defined as a straight line, where the activation is proportional to the input i.e. the weighted sum from neurons. Hence, it is mathematically defined as:

$$f(x) = ax + c \tag{2.2}$$

Linear activation function has a lot of drawbacks and are not usually used in modern artificial neural networks. Specifically, applying this function in all the nodes makes the activation function work like linear regression. The final layer of the Neural Network will be acting as a linear function of the first layer. Another issue is that the output can not be limited in a specific range.

**Non-Linear Activation Functions**: Modern artificial neural networks mostly use Non-Linear Activation Functions, the most popular of which are the following:

- **Sigmoid Activation Functions**: This function takes as an input a real value ranged in $(-\infty, \infty)$ and outputs another value between 0 and 1. It is used mostly when the output has to be a probability and is defined as:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.3}$$

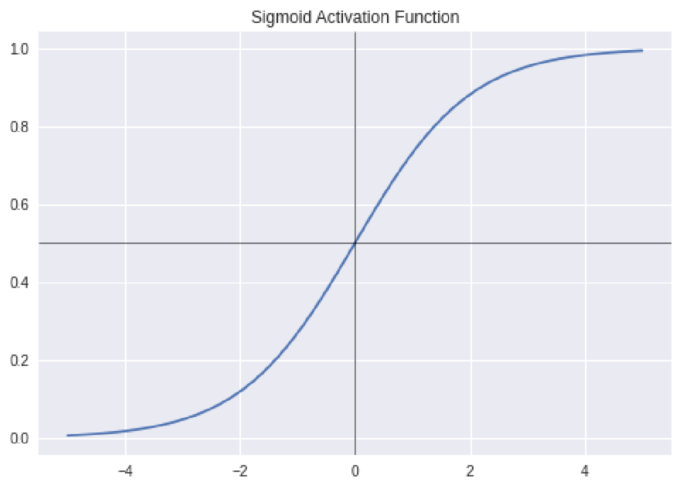


**Figure 2.3.** *The sigmoid function*

- **Tanh Activation Functions**: This function maps values between -1 and 1. Similar to the sigmoid function, the derivative of tanh can be expressed in terms of the function itself. It is defined as the ratio of the hyperbolic sine and hyperbolic cosine functions and can therefore be defined as:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x e^{-x}} \tag{2.4}$$

- **Rectified Linear Unit (ReLU)** Activation Functions: ReLU is a non-linear function and is the most commonly used activation function in neural networks. It returns the value of the input, or 0, if the input value is negative. Hence, its mathematical definition is:

$$f(x) = max(x, 0) \tag{2.5}$$

ReLU activation function has also a few variants that alleviate some of its problems:

  - **Leaky ReLU**: This function is a linear variant of ReLU. Instead of being 0 when z<0, a leaky ReLU allows a small, non-zero, constant gradient α (usually, α=0.01). Leaky

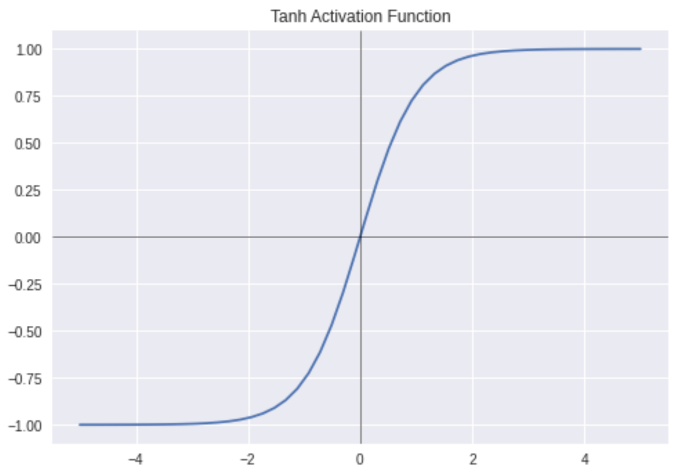


**Figure 2.4.** *The tanh function*

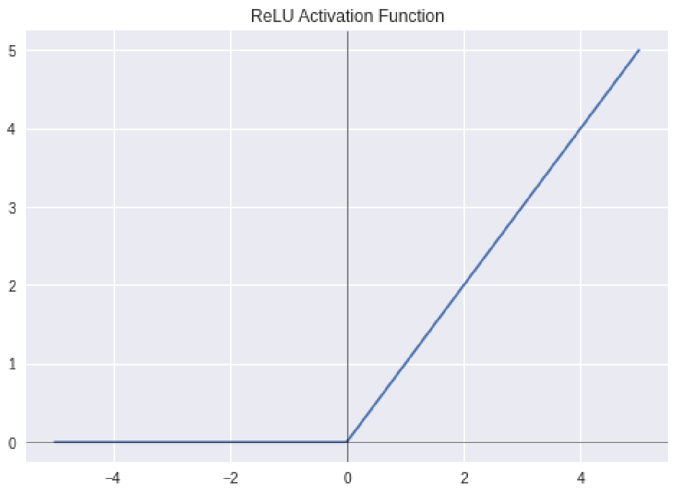


**Figure 2.5.** *The ReLU function*

ReLUs attempt to fix the “dying ReLU” problem. It is defined as:

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ 0.01x & \text{otherwise.} \end{cases} \tag{2.6}$$

– **ELU**: Exponential linear units try to make the mean activations closer to zero, which speeds up learning. It has been shown that ELUs can obtain higher classification accuracy than ReLUs. This activation function is defined as:

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ \alpha(e^x - 1) & \text{otherwise.} \end{cases} \tag{2.7}$$

### 2.3.2 Loss Function

As described above, every Supervised Learning algorithm aims to create a mapping function *f()*, that corresponds an input to the appropriate label. In order the model to be able to learn this

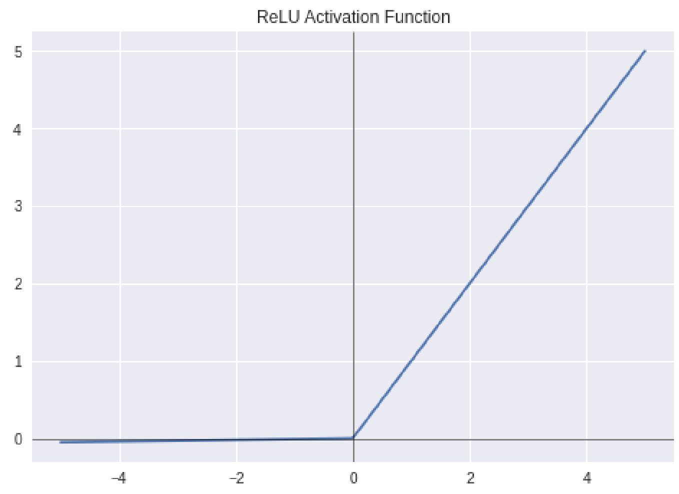


**Figure 2.6.** *The Leaky ReLU function*

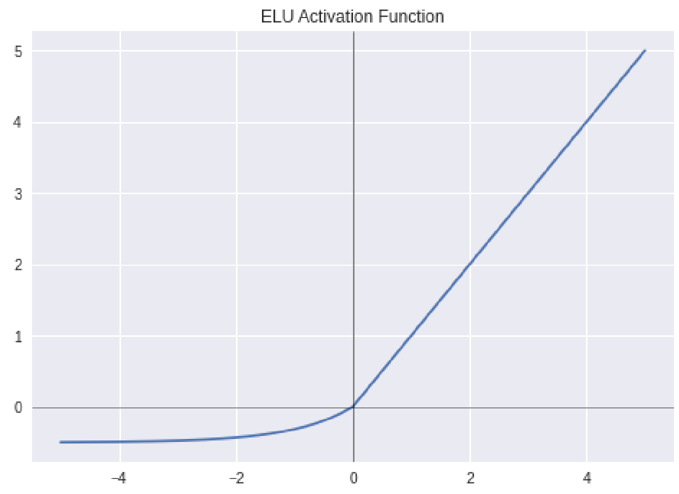


**Figure 2.7.** *The ELU function*

function, a method is needed to compute the distance (loss) between the output of the algorithm after each epoch and the expected output. To compute the loss of the model, a Cost Function $L(\hat{y}, y)$ is defined , where $\hat{y}$ is the predicted label and $y$ the actual label. The goal is to minimize this function during the training.

Given a train set $(x_{1:n}, y_{1:n})$, a cost function $L$ per sample and a function $f(x;\theta)$, the total loss is defined as the average loss on all training data:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}\left(f(x;\theta), y_i\right) \tag{2.8}$$

The goal is to find the optimal parameters $\theta$ that minimize the total error:

$$\hat{\theta} = \arg_\theta \min \mathcal{L}(\theta) = \arg_\theta \min \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}\left(f(x;\theta), y_i\right) \tag{2.9}$$

Depending on the task or the dataset, a different loss function should be selected, since classification models (binary or multi-label) and regression ones should use a different method to calculate the loss. A few standard cost functions are presented below:

**Mean Squared Error (MSE):** MSE calculates the mean squared prediction error:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - P_i)^2 \tag{2.10}$$

Where the prediction error is the difference between the true value $(Y_i)$ and the predicted value $(P_i)$ for an instance, and $\partial$ is the parameter vector of the network. MSE is used with regression models.

**Mean Absolute Error (MAE):** MAE calculates the mean of the absolute prediction error:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}|Y_i - P_i| \tag{2.11}$$

Where $Y_i$ is the true value and $P_i$ is the predicted value for an instance, and $\partial$ is the parameter vector of the network.

**Cross Entropy:** Cross-entropy loss function uses the concept of cross-entropy. Cross-entropy is mathematically defined as:

$$H(p, q) = -\sum_{k} p_k \log q_k \tag{2.12}$$

Where $p$ and $q$ are the true and the predicted probability distributions, respectively, the more the two distributions differ, the higher the value of the cross-entropy. The cross-entropy loss function is mostly utilized in classification problems. Using this function, the model aims to minimize the cross-entropy between the model's distribution and the distribution of the given data.

### 2.3.3 Gradient descent

After defining the loss function, a method is needed to minimize it. Gradient-based methods have been used for this purpose, as they utilize gradients' property to point to the direction of the most significant increase of the given function. These methods minimize the objective function $\mathcal{L}(\theta)$ by repeatedly computing an estimate of the loss $\mathcal{L}$ over the training set, computing the gradients of the parameters $\theta$ of the model concerning the loss estimate and updating the parameters in the opposite direction of the gradient.

The most popular gradient-based algorithms are:

*Gradient descent* (GD), which computes the gradient of the cost function concerning the parameters $\theta$ for the entire dataset. GD's most important hyperparameter is the *learning rate* $(\eta)$ as it controls the extent to which the model parameters are adjusted concerning the loss gradient. GD is formally defined as:

$$\theta = \theta - \eta \nabla_\theta J(\theta) \tag{2.13}$$

*Stochastic Gradient Descent (SGD)*, which instead of computing the gradients using the entire dataset, it performs a parameter update for each training example $\mathbf{x}_i$ and label $y_i$ :

$$\theta = \theta - \eta \nabla_\theta J\left(\theta; \mathbf{x}_i; y_i\right) \tag{2.14}$$

A significant advantage of the Stochastic Gradient Descent against the Gradient descent is that it is usually much faster, since the latter performs redundant computations for large datasets, as it recomputes gradients for similar examples before each parameter update, while the former performs one update at a time.

The selection of the learning rate hyperparameter is also a crucial point. Opting for a small value may lead to slow convergence, while selecting a very large one might cause the model to fail to converge or even diverge.

### 2.3.4 Backpropagation

Towards minimizing the cost function in an artificial neural network, using an optimum set of values for the parameters $\vartheta$ (weights), we have to compute the gradient. This procedure ends up very complex and innefficient when using bigger and mor ecomplicated networks, even if the the chain rule is used for the computation. To tackle this problem, [31], [32] suggested computing the gradients through backpropagation algorithm . Backpropagation computes the derivatives of a complex mathematical relation, using the chain rule and saving the results in the interim. The goal is to update the weights of the network in the optimal way after each computation of the loss function. The computation is performed for one layer of the network at a time, starting from the last layer and iterating backwards. The gradients computed can help us understand how quickly the loss function changes when the weights change and thus how well the network is performing. In this way, it is easier to fine tune the weights so as to further minimize the model's loss and improve the overall performance.

### 2.3.5 Generalization

A machine learning algorithm is expected to train a model not only to perform well on data seen during the training process, but also to be able to handle new, unseen data. This is due to the fact that the data that has been collected and used for the training is only a sample of all the possible inputs, hence it is incomplete and probably noisy. Generalization is a crucial concept in machine learning and refers to a model's ability to adapt properly and handle new, previously unseen data, drawn from the same distribution as the one used to create the model. In other words, generalization evaluates a model's ability to process new data and generate accurate predictions after being trained on a fixed training set.

During the training process, the model computes an error based on how it performed on the training set, called training error and reduces this error while the training continues. Instead of just optimizing the model's behaviour only the training set, we have to take into account the test (generalization) error. Typically this error is estimated by measuring the model's performance on a test set of examples that were collected separately from the training set.

Determining if a machine learning algorithm is successful can be based on how the model handles data seen during the training process as well as how it adopts to unseen data. Based on the above paragraph the two factors can be expressed as how small is the training error and how small is the gap between training and test error respectively and represent two key concepts of machine learning;

*Underfitting*, which occurs when the model was not able to learn the training data nor generalize to new data, thus leading to sufficiently low error value on the training set.

*Overfitting*, which occurs when the model has learned the training data too well. This means that the model learns every detail and even noise present in the training data to the extent that it negatively impacts the performance of the model on new data, since it can not adapt properly, searching for very specific patterns in the new data. Overfitting will lead to a significant gap between the training error and test error.

A crucial goal while training a machine learning model is to find a good trade-off of training error and the gap between training and test error. This way the model will have learned sufficiently well the training data, but will still be flexible to handle unseen data.

### 2.3.6 Tackling overfitting

The problem of overfitting is one of the most crucial challenges, while training a model. There have been proposed, although, a few techniques that have been proven effective and have been widely used through the years, improving the generalization.

**Regularization** is the most common technique to alleviate the problem of overfitting. In order to enforce generalization, restrictions are imposed on the form of the solution by forcing the model to choose the smallest - in order of parameters- solution. This is done by implying a term in the loss equation and then penalizing the size of the model. Thus, the loss function takes the following form:

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta) = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L\left(f\left(\mathbf{x}_i ; \theta\right), y_i\right) + \lambda R(\theta) \tag{2.15}$$

The regularization term considers the parameter values and scores their complexity. Regularization inherently aims to penalize complex models and favor simpler ones, looking for values that have both a low loss and low complexity. An important hyperparameter that comes with this method is $\lambda$; a value that is set manually, based on the classification performance on a development set. In the above equation, $R$ stands for the regularizer. Regularizers measure the norms of the parameter matrices and opt for solutions with low norms. The two most common regularization norms are $L_2$ and $L_1$.

$L_1$ **regularization.** The $L_1$ regularizer, also called a sparse prior or lasso, encourages sparse solutions or models with many parameters with a zero value. To do so, it punishes uniformly low and high values and intends to decrease all non-zero parameter values towards zero.

$$R_{L_1}(\mathbf{W}) = \|\mathbf{W}\|_1 = \sum_{i,j} \left|\mathbf{W}_{[i,j]}\right| \tag{2.16}$$

$L_2$ **regularization.** $R$ takes the form of the standard Euclidean norm ($L_2$-norm) of the parameters, trying to keep the sum of the squares of the parameter values low. Large model weights $\mathbf{W}_{[i,j]}$ will be penalized, since they are considered "unlikely". $L_2$ is often referred to as weight decay. As one can observe, high weights are severely penalized, but weights with small values are only negligibly affected.

$$R_{L_2}(\mathbf{W}) = \|\mathbf{W}\|_2^2 = \sum_{i,j} \left(\mathbf{W}_{[i,j]}\right)^2 \tag{2.17}$$

**Dropout**: This method is designed to prevent the network from learning to rely on specific weights. It randomly sets to zero (drops) a percentage of the neurons in the network in each training example, during the stochastic-gradient training. This technique is one of the key factors contributing to robust results of neural networks.

**Pruning** Pruning is the technique of removing connections between neurons or entire neurons, channels, or filters from a trained network, which is done by zeroing out values in its weights matrix or removing groups of weights entirely.

### 2.3.7 Machine Learning Models

In this section a few basic machine learning classification models will be presented.

### Logistic Regression

Logistic regression (LR) is a linear classification model, which computes the probabilities for a classification problems with two possible outcomes by applying the logistic function to the output of a linear function $f$ . The logistic function, also known as the sigmoid function, squeezes a vector into a range of (0, 1). For a binary classification problem, the probability of one of the classes for a feature vector $x \epsilon \mathbf{R}^d$ is defined as:

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-f(\mathbf{x})}} \tag{2.18}$$

where $f$ is a linear function with $w_d$ parameters:

$$f(\mathbf{x}) = w_0 + w_1 x_1 + \cdots + w_d x_d \tag{2.19}$$

Since we examine problems with only two classes, the probability of the other class can be computed as:

$$P(y = 0|\mathbf{x}) = 1 - P(y = 1|\mathbf{x}) \tag{2.20}$$

The parameters of the linear function are computed by minimizing the cross-entropy loss $J$, which is defined as:

$$J(\mathbf{w}) = -[ylog(P(y = 1|\mathbf{x})) + (1 - y)log(P(y = 1|\mathbf{x}))] \tag{2.21}$$

### Support Vector Machines

Support Vector Machines (SVMs) are a set of linear supervised learning methods used for classification, regression and outliers detection. An SVM is an algorithm for maximizing a particular mathematical function with respect to a given collection of data. Assuming a two-class classification problem using a linear model of the form:

$$f(\mathbf{x} = \mathbf{w}^T \phi(\mathbf{x}) + b \tag{2.22}$$

where $\phi(\mathbf{x})$ denotes a fixed feature-space transformation and b is a bias parameter. Additionally, we assume for the moment that the training data set is linearly separable in feature space. The training dataset consists of $N$ input vectors $x_1, x_2, \cdots, x_N$, with corresponding target values (classes) $y_1, y_2, \cdots, y_N$, where $y_i \epsilon \{-1, 1\}$. The new data points $x$ are classified according to the sign of $f(\mathbf{x})$, for which based on the assumption of the linearly separable dataset, there exists at least one choice of the parameters w and b such that a function of the above form satisfies $f(\mathbf{x}) > 0$ for points having $y_i = +1$ and $f(\mathbf{x}) < 0$ for points having $y_i = -1$, so that $y_i f(\mathbf{x}) > 0$ for all training points.

The separating hyperplane is a hyperplane that separates the classes of our problem. In the case of the two classes, the separating hyperplane is a linear function that distinguishes the two classes. As shown in Fig. 2.8 there can exist numerous (infinite) seperating hyperplanes in a problem, thus SVM's purpose is to find the optimal hyperplane for which the minimum distance between the classes is as wide as possible. This final hyperline is depicted by the maximum-margin hyperplane.

## 2.4 Deep Learning Models

The research on field of Artificial Intelligence through the last years has mostly focused on Deep Learning models and has achieved unprecedented results in a wide variety of problems paving the

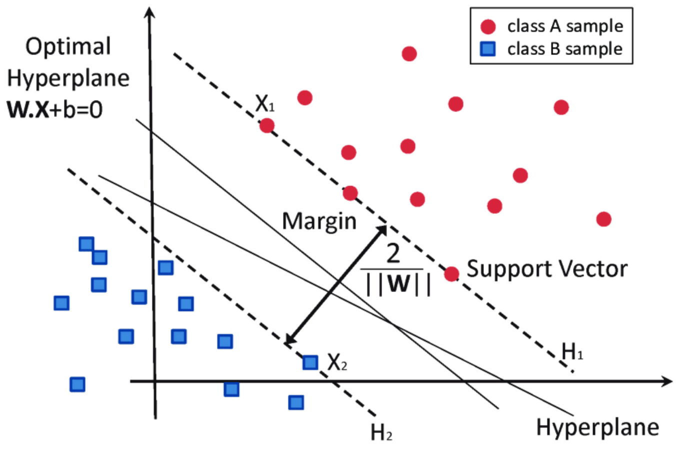


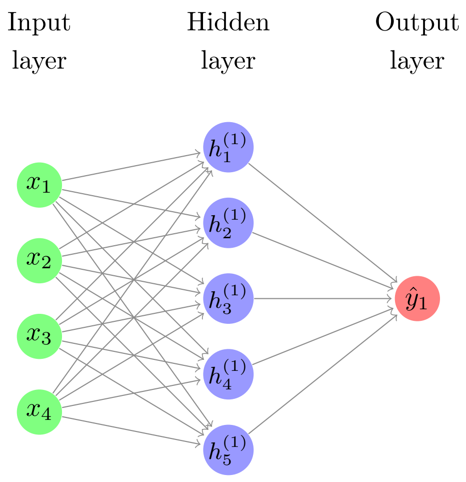
**Figure 2.8.** *Classification of data by support vector machine (SVM). Source: [1]*

road for fields such as natural language processing, computer vision and bioinformatics as well as reasoning and artificial general intelligence (AGI). Deep Learning models are consisted of neural networks stacked together in multiple levels. Their primary goal is to extract high-level features from raw data by propagating the input through the consecutive levels of such architectures, each of those transforms the input into a different representation.

### 2.4.1 Feedforward Neural Networks

An **Artificial Neural Network (ANN)** is a computational model inspired by the network of neurons present in the human brain. Each neuron, brain's computational unit, receives input signals from its dendrites and produces output signals along its (single) axon. The axon connects to the dendrites of other neurons using synapses. The core idea is that all the signals that travel along the axon interact with the dendrites of the other neurons based on the *learnable* synaptic strength (weight) at that synapse, which controls the impact one neuron has on another one. The signal is transferred, through the dendrites, to the cell body, where the summation takes place. This final sum will determine whether the specific neuron will be activated. Specifically, if a certain threshold is surpassed, then the neuron fires sending a spike along its axon. (In the below figure, a biological neuron and its mathematical notation is presented:)

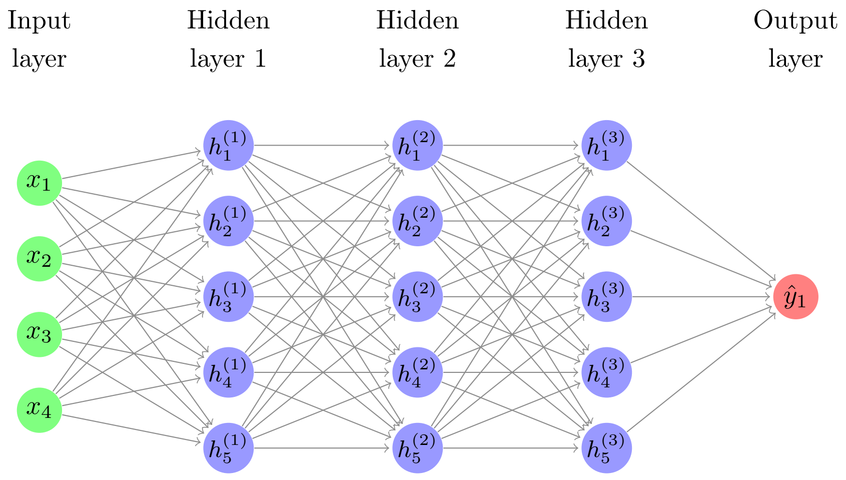
A Feed Forward Neural Network is an artificial neural network in which the connections between nodes do not form a cycle. In these architectures, also often called Multilayer Perceptrons (MLPs), the information is only processed in one direction from the input nodes, through the hidden nodes to the output nodes. In each node of an FFNN the weighted sum of its' inputs is usually computed , followed by a non-linear activation function. In its simplest form, the network consists of three layers: the input layer, the hidden layer, and the output layer. This type of network is called *single-layer perceptron* network and it is shown below:



Multilayer Perceptrons are constructed by stacking multiple FFNNs together. In each layer, all neurons have directed connections to the neurons of the subsequent layer. The multiple layers and the non-linear activation give MLPs the ability to distinguish data that is not linearly separable. More specifically, MLPs are composed of the following layers:

- **Input layer**: Accepts the input data and passes it to the hidden layer without any further computation.
- **Hidden layer(s)**: At least one hidden layer preprocesses the inputs obtained by the previous layer, extracting the required features from the input data. Higher hidden layers extract higher-level features.
- **Output layer**: The last layer that receives the processed data, generating output in order to conclude to a decision (e.g. output logits for classification)

An example of Multilayer Perceptron structure with 3 hidden layers is:



### 2.4.2 Convolutional Neural Networks (CNN)

Convolutional Neural Networks [33] were proposed as a regularized versions of multilayer perceptrons and gained a lot of research attention, since they were proven very successful on a wide variety of applications, especially when used on images. Motivated by the connectivity pattern between neurons in animal's visual cortex [34], neurons in CNNs receive input from a corresponding neighborhood of the previous layer, while they share weights in each layer. These architectures were able to alleviate major multilayer perceptron's problems: (i) prone to overfitting, (ii) computationally intensive.

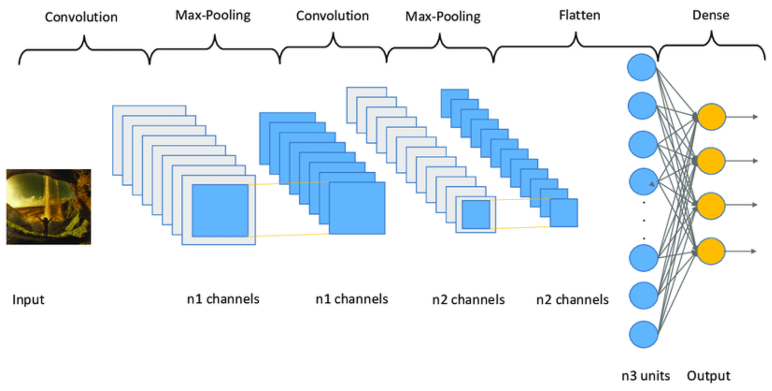


**Figure 2.9.** *Typical CNN architecture. Source: researchgate.net*

The core idea of Convolutional Neural Networks is to exploit hierarchical patterns in data and assemble more complex patterns using smaller and simpler patterns. The building blocks of CNNs are:

- **Convolutional layer**: Consists of a set of learnable filters , which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the both dimensions of the input, computing the dot product between the filter entries and the input and generating a 2-dimensional activation map of that filter. Subsequently, the network is able to learns filters that activate, when it detects some specific type of feature at some spatial position in the input.
- **Pooling layer**: A form of non-linear down-sampling. Its purpose it to progressively reduce the spatial size of the representation and as a result the number of parameters, while it also controls overfitting. There exist several pooling functions, with the most popular to be the max pooling, which splits the the input into a set of rectangles and, for each such sub-region, outputs the maximum.
- **Fully connected layer**: The final classification is done via fully connected layers, in order to learn non-linear combinations of the high-level features.

### 2.4.3 Recurrent Neural Networks

Recurrent Neural Networks came to address one significant disadvantage of Feedforward Neural Networks, which prevent them from achieving great results mostly in sequential data. Specifically, in FFNNs each part of an input sequence is processed independently, hence it is impossible for the model to consider the context in which every part is found. A recurrent neural network (RNN) [35], proposed in the 1980s, is a type of artificial neural network where connections between units form a directed cycle. This characteristic of RNNs creates an internal state of the network, which enables it to exhibit dynamic behavior. Unlike feedforward neural networks, RNNs are able to utilize their internal memory to process and work on arbitrary sequences of inputs. This advantage has proven RNNs very effective against tasks such as speech recognition, Natural Language Processing and Machine Translation.

RNNs are especially useful with sequential data because each neuron can use its internal memory to maintain information about the previous input. Specifically, they store the context, up to step $t$, in a hidden state vector, denoted with $s_t$ . At each new time-step, this state vector gets updated to also reflect the latest part of the input sequence. The output $o_t$ for the RNN is a function of it's input $x_t$ and the previous state $s_{t1}$. As shown in Fig. 2.10, a recurrent neural network can be described as multiple copies of the same network, each passing a message to a successor.

A vanilla RNN first recieves the $x_0$ from the sequence of input, and outputs the hidden state $h_0$. The hidden state $h_0$ along with the next input $x_1$ is the input for the next step. Accordingly, $h_1$ along with $x_2$ is the input for the next step and so on. Using this method, the RNN is able to "remember" the context of the input it has already seen while training. This characteristic allows RNN to distinguish between two sentences containing the same words, but with different order, which gives the sentence a total different meaning. For example RNN is able to grasp the differences between the sentences *"I had my hair cut"* and *"I had cut my hair"*.

Formally, at each time step $t$ a vanilla Recurrent Neural Network is defined as follows:

$$h_t = \sigma_h(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \tag{2.23}$$

$$y_t = \sigma_y(W_{yh}h_t + b_y) \tag{2.24}$$

where $h_t$ is the hidden state at time step t, $x_t$ is the input vector at time step $t$, $y_t$ is the output vector at time step $t$, $b_h$ is the bias for h, $b_y$ is the bias for y and $\sigma_h$ and $\sigma_y$ are activation functions. Lastly, the weights are three separate matrices: $W_{hx}$ (input-to-hidden weights), $W_{hh}$ (hidden-to-hidden), and $W_{yh}$ (hidden-to-output).

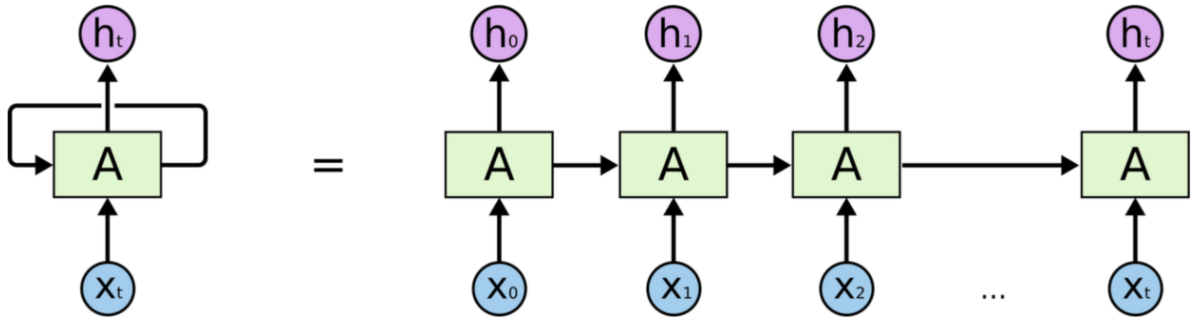


**Figure 2.10.** *A recurrent neural network unfolded for a sequence of length t. Source: data-science.eu*

While RNNs were very promising in handling sequential data, in practice they fall short when it comes to large contexts. For example, RNNs may perform efficiently predicting a masked word in a small sequence of words, such as: *"It is a sunny summer day and the temperature is [MASK]"*, but as the sequence gets bigger and the number of words after the most important ones grow, it becomes impossible for the RNN to "remember" all that.

### Long Short Term Memory (LSTM) Networks

The incapability of RNNs to capture long-range dependencies has motivated a lot of research towards this direction, with the Long Short Term Memory Networks [36] being maybe the most successful approach. LSTMs are a special kind of RNN, designed for and capable of learning long-term dependencies. As they are derived from conventional RNNs, they preserve the concept of hidden states while also add one more concept; the cell state. These networks have the ability to add or remove information to the cell state, utilizing mechanisms called LSTM gates. In Fig. **??** a visualization of the LSTM network is shown:

Formally, assuming a sequence $x_1, x_2, \cdots, x_t, \cdots, x_n$ of vectors of an input sequence of length n, for a vector $x_t$ , with inputs $h_{t1}$ and $c_{t1}$, the hidden-state $h_t$ and cell state with $C_t$ for time-step t are computed as follows:

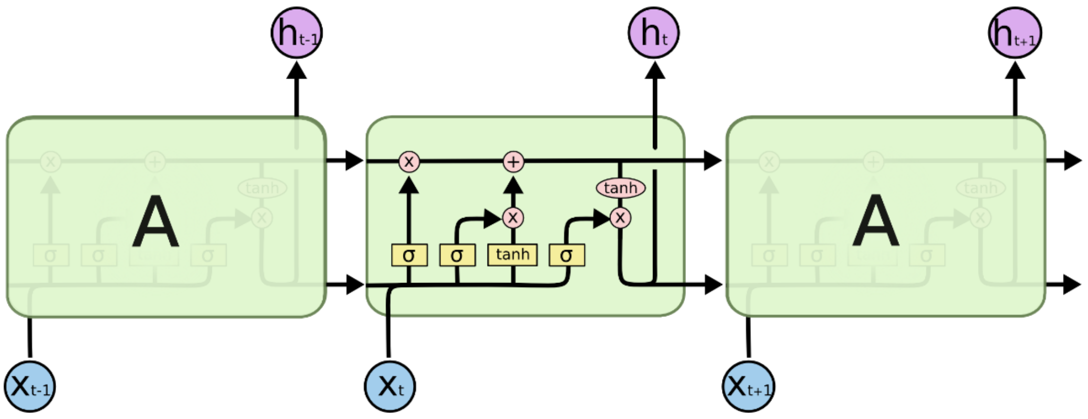


**Figure 2.11.** *The repeating module in an LSTM. Source: colah.github.io*

$$
\begin{aligned}
\mathbf{f}_t &= \sigma\left(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f\right)\\
\mathbf{i}_t &= \sigma\left(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i\right)\\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o\right)\\
\mathbf{u}_t &= \tanh\left(\mathbf{W}_u\mathbf{x}_t + \mathbf{U}_u\mathbf{h}_{t-1} + \mathbf{b}_u\right)\\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{u}_t\\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right)
\end{aligned}
\tag{2.25}
$$

The functions of the three gates and of the cell state are described below:

**Forget gate** ($f_t$)**.** This gate determines which information should be preserved or discarded. Information from the previous hidden state $h_{t-1}$ together with information from the current input $x_t$ is passed through a sigmoid activation function, which squeezes the values between 0 and 1. Since the result is then being used as a coefficient, the closer to 0 means to forget, and the closer to 1 means to keep.

**Input gate** ($i_t$)**.** The previous hidden state together with the current input is passed into a sigmoid function, to squeeze the values between 0 and 1 and determine which values will be updated (0 means unimportant and 1 means important). The hidden state and current input are also passed to the tanh function to squish values between $-1$ and $1$ ($u_t$). Finally, the tanh output is multiplied with the sigmoid output ($i_t \odot u_t$), so that the latter will filter the important information of the former.

**Cell state** ($c_t$)**.** To compute the next cell state, firstly the current cell state $c_t$ gets pointwise multiplied by the result of the forget gate. This results in dropping information from the cell state that is not that important. Then, a pointwise addition is applied between previous result and the output from the input gate, that updates the cell state to new values that the neural network finds relevant.

**Output gate** ($o_t$)**.** The output gate determines the next hidden state. As the hidden state contains information from previous inputs, it is also used for predictions. After, passing the previous hidden state and the current input into a sigmoid function, the newly modified cell state is passed to the tanh function. We multiply the tanh output with the sigmoid output ($o_t \odot \tanh(\mathbf{c}_t)$) to decide what information the hidden state should carry. The output is the hidden state. The new cell state and the new hidden is then carried over to the next time step.

Chapter 3

# Graph Neural Networks

## 3.1 Introduction

Graphs are a type of data structure, which is used to model a set of objects, represented as the nodes of the graph, and their relationships, the graph edges. Through the last years, researches of analyzing graphs with machine learning have been receiving increasing attention because of the great expressive power of graphs. Specifically, graphs can be used to represent a large number of systems across various areas, including social science (social networks) [13], [14], natural science (physical systems [15], [16] and protein-protein interaction networks [17]), knowledge graphs [18] and many other research areas [19]. Graph neural networks [37] are deep learning based methods that operate on graph domain. Because of their impressive performance and high interpretability, GNNs have seen through the last years increasing popularity and have been widely applied for graph analysis methods, which focus among others on node classification, link prediction, and clustering.

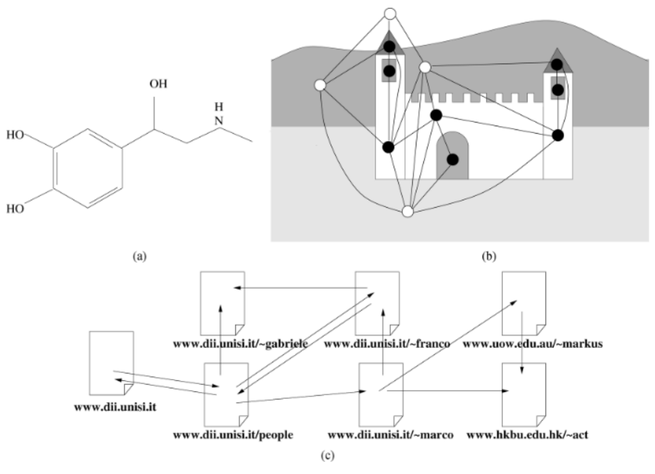


**Figure 3.1.** *Some applications where the information is represented by graphs: (a) a chemical compound (adrenaline), (b) an image, and (c) a subset of the web. Source: [2]*

The motivation behind Graph Neural Networks originates from convolutional neural networks (CNNs)[38]. CNNs achieved exceptional results and led to breakthroughs in almost all machine learning areas, thanks to their ability to extract multi-scale localized spatial features and compose them to construct highly expressive representations. CNNs have some key features, that are also of crucial importance in facing problems of graph domain. These features are 1)local connections 2) shared weights and 3) the use of multi-layer. However, CNNs are by definition designed to handle single-dimensional (text) or two-dimensional data (images). This led to the need of having a more generalized model, that can handle data that are represented in non-Euclidean domain. Another factor that also motivated the generation of Graph Neural Networks, were the graph embeddings [39], [40], that enable learning to represent graph nodes, edges or subgraphs in low-dimensional vectors.

Based on CNNs and graph embedding, graph neural networks (GNNs) are proposed to collectively aggregate information from graph structure. The aggregation that takes place is invariant of the order of the graph nodes. In contrast to standard neural networks, such as CNNs and RNNs, which stack the feature nodes by a specific order, GNNs respect the fact that there isn't a natural order of the nodes in the graph. To achieve an traverse invariant aggregation the traditional neural networks should traverse the input nodes in all possible ways, something that would be lead to very inefficient training. Another significant advantage of GNNs is that the graph edges represent the dependency between two nodes. The network can then perform the propagation guided by the graph structure. Lastly, GNNs have the potential to lead to breakthrough achievements on reasoning, which is a very crucial research topic for high-level artificial intelligence and a task that standard neural networks fall short. GNNs generate the graph from non-structural data, such as pictures and documents, which can then be utilized for the reasoning.

## 3.2 Characteristics of GNNs

### 3.2.1 Permutation invariance and equivariance

One core principle of Graph Neural Networks is that they do not depend on the order of the input graph's nodes. As a result, we can say that they do not operate on a *list* of nodes, rather on a *set* of nodes. Assuming a graph G containing only nodes **x** (ignoring the edges for now) and defining $x_i \epsilon \mathbb{R}^k$ as the features of node $i$, we can get the node feature matrix of shape $n$ as:

$$X = [x_1, \cdots x_n]^T \tag{3.1}$$

Stacking, however, the nodes' features defines inevitably a node ordering. As a result, the goal is to learn a function that will not depend on that order, hence it will be *invariant* to any permutation on these nodes. In linear algebra, permutation matrices are $nxn$ matrices that apply a permutation of the rows of the matrix they are left-multiplied with. These matrices have exactly one 1 in every row and column, and zeros everywhere else. A typical example of a permutation matrix **P** left-multiplied with a matrix **X** is shown below:

$$P_{(2,4,1,3)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} - - x_1 - - \\ - - x_2 - - \\ - - x_3 - - \\ - - x_4 - - \end{bmatrix} = \begin{bmatrix} - - x_2 - - \\ - - x_4 - - \\ - - x_1 - - \\ - - x_3 - - \end{bmatrix}$$

(3.2)

In order to ensure that the result will not depend to the order of the nodes, the concept of **permutation invariance** should apply.Specifically, a function **f** should be applied to the input features **X**, such that **f(X)** will be permutation invariant. **f(X)** is permutation invariant if, for all permutation matrices **P**:

$$f(PX) = F(X) \tag{3.3}$$

The notion of permutation invariance, however, destroys any node-level information, since the nodes are all grouped together. Hence, permutation-invariant models are a sufficient way to obtain only set-level outputs. Nevertheless, in practice node outputs are in many cases very valuable, which leads to the need of using functions that do not change the node order, but do return the same result if the permutation happens before or after the function. The above lead us to the concept of **permutation equivariance**. Accordingly, f(X) is permutation equivariant if, for all permutation matrices P:

$$f(PX) = Pf(X) \tag{3.4}$$

After defining the above for simple graphs, containing only nodes, we now have to extend the above equations for actual graphs, taking into account the edges connecting the graph's nodes.

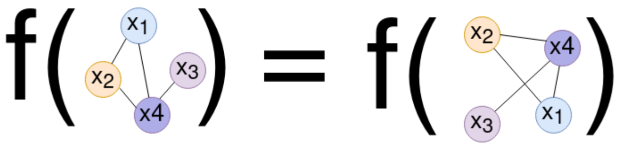


**Figure 3.2.** *Permutation invariance*

Specifically, in order to augment the set of nodes V with edges, we consider a set of edges $E \subseteq VxV$ and represent them using a simple adjacency matrix A, such that:

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j)\epsilon E, \\ 0 & \text{otherwise.} \end{cases}$$

(3.5)

A permutation to the adjacency matrix **A** using a permutation matrix **P** can be achieved as $PAP^T$. Finally, the suitable functions f(X,A) to be applied over the graphs, satisfying the permutation invariance and equivariance respectively are shown below:

$$\textbf{invariance: } f(PX, PAP^T) = f(X, A) \tag{3.6}$$

$$\textbf{equivariance: } f(PX, PAP^T) = Pf(X, A) \tag{3.7}$$

### 3.2.2 Locality on graphs

One core characteristic of graphs is that they express locality, through the concept of the *neighbourhood*. Specifically, given a graph $G = (V, E)$ we can define for each node $i$ its (1-hop) neighbourhood as:

$$N_i = j : (i,j)\epsilon E \vee (j,i)\epsilon E \tag{3.8}$$

The above definition ignores the direction of the edges. Typically, it is also assumed that $i\epsilon N_i$. Based on the above, a multiset of features within each node $i$'s neighbourhood can be obtained as:

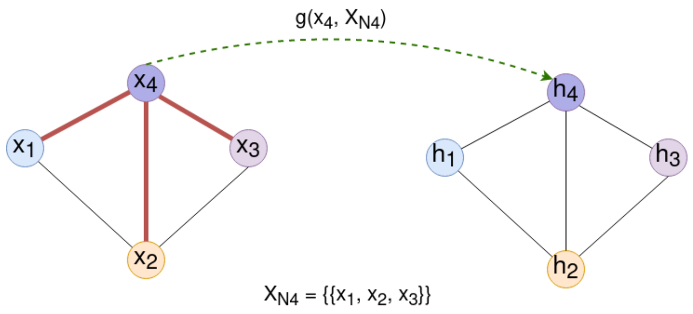
$$X_{N_i} = x_j : j\epsilon N_i \tag{3.9}$$

A local result for each output can be obtained by applying a *local* function g over each mutliset: $G(x_i, X_{N_i})$

Finally, by appropriately applying the local function, g, over all neighbourhoods, we can construct permutation equivariant function, f(X, A) as:

$$F(X, A) = \begin{bmatrix} - - g(x_1, X_{N_1}) - - \\ - - g(x_2, X_{N_2}) - - \\ \vdots \\ - - g(x_n, X_{N_n}) - - \end{bmatrix} \tag{3.10}$$

Based on all of the above for the functions $f$ and $g$, a permutation equivariant function and a permutation invariant one should be respectively selected.



In the next sections we present a set of general Graph Neural Network frameworks, as well as popular models that are being widely used for a lot of tasks.

## 3.3 Models

In this section we review various popular graph neural network models. We first present the original framework and its limitations and then its variations. Before digging into the details, we provide the notations that will be used in the rest of the thesis, regarding the GNNs.

### 3.3.1 Graph Neural Networks

Here the original Graph Neural Network framework is presented as it was proposed in [2], which enabled to feed in a neural network non-Euclidean structures, represented as graphs. As defined in graph theory, a graph is composed by as set of nodes **V** and a set of edges **E**. Each node is described by its features and its neighbouring nodes, i.e. the information contained in its neighbourhood. GNNs aim to learn a state embedding $h_v \epsilon \mathbb{R}^s$ for each node, which will contain the information of its neighbourhood. This state embedding , $h_v$ of the node v can later be used to compute an output $o_v$.

In all GNN frameworks exist two main functions:

1. $\mathbf{f_w}$ parametric function, called local transition function that expresses the dependence of a node on its neighborhood. This function updates the node state, according to each neighbourhood, as:

$$\mathbf{h_v} = f(\mathbf{x_v}, \mathbf{x_{co[v]}}, \mathbf{h_{ne[v]}}, \mathbf{x_{ne[v]}}), \tag{3.11}$$

where $x_v, x_{co[v]}, h_{ne[v]}, x_{ne[v]}$ are the features of v , the features of its edges, the states, and the features of the nodes in the neighborhood of v , respectively.

2. $\mathbf{g_w}$, called local output function that describes how the output is produced:

$$\mathbf{o_v} = g(\mathbf{h_v}, \mathbf{x_v}), \tag{3.12}$$

By stacking all the states, all the outputs, all the features, and all the node features respectively, we get $\mathbf{H}, \mathbf{O}, \mathbf{X}, \mathbf{X_N}$ and conclude to the compact form:

$$\mathbf{H} = \mathbf{F}(\mathbf{H}, \mathbf{X}), \tag{3.13}$$

$$\mathbf{O} = \mathbf{G}(\mathbf{H}, \mathbf{X_N}) \tag{3.14}$$

For the (t+1)-th iteration the states are computed as:

$$\mathbf{H^{t+1}} = \mathbf{F}(\mathbf{H^t}, \mathbf{X}) \tag{3.15}$$

In supervised learning, the parameters of $f$ and $g$ are learned, according to target information ($t_v$ for node v) based on the loss:

$$loss = \sum_{i=1}^{p}(t_i - o_i) \tag{3.16}$$

where p is the number of supervised nodes.

Based on the gradient-descent strategy, the learning algorithm can be described in three steps: **1)** Iteratively update the states $h_v^t$ by Eq. 1 until a time T, when they approach the fixed point solution of Eq. 3: $H(T)H$, **2)** Compute the gradient of weights W, using the loss, **3)** Update the weights W according to the gradient computed in the last step.

The original GNN suffers from several **limitations**:

1. Original GNNs are unable to effectively model informative features, that might be present in the edges of the graph.
2. Unlike most popular neural networks, which use different parameters in each iteration, GNN uses the same parameters.
3. It is inefficient to update the hidden states of nodes iteratively for the fixed point. If relaxing the assumption of the fixed point, we can design a multi-layer GNN to get a stable representation of node and its neighborhood.
4. It is unsuitable to use the fixed points, if we focus on the representation of nodes instead of graphs because the distribution of representation in the fixed point will be much smooth in value and less informative for distinguishing each node.
5. Lastly, the update of node hidden states is a sequential process, which can benefit from RNN based frameworks, like GRU and LSTM[41].

## 3.3.2 General Graph Neural Network Frameworks

In addition to several graph neural networks variants that will be discussed later, a few general frameworks have been proposed intending to integrate different models into one single framework.

#### Message Passing Neural Networks

[28] introduced a general framework, to be used for supervised learning on graphs called Message Passing Neural Networks (MPNNs). This framework provides an abstract model that preserves the common characteristics between major successful GNN models for graph structured data.

The framework is consisted of two phases; the message passing phase and a readout phase. The former's purpose is to aggregate the information (messages) from each node's neighbourhood and update its hidden state. Specifically, two functions, the message function $M_t$ and vertex update function $U_t$ are defined and the messages $m_v^t$ and the updated hidden states $h_v^{t+1}$ are computed as follows:

$$m_v^{t+1} = \sum_{w \epsilon N_v} M_t(h_v^t, h_w^t, e_{vw}) \tag{3.17}$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \tag{3.18}$$

where $e_{vw}$ are the features of the edge from node $v$ to node $w$.

The readout phase aims to generate a graph feature vector using a readout function $R$, as it is described below:

$$\hat{y} = R(\{h_v^T | v \epsilon G\}) \tag{3.19}$$

The definition of the three functions used in the above equations, $M_t, U_t$ and $R$ could have different specific settings and given the abstract profile of the framework, the MPNN could generalize several different models via different function settings.

### Non-local Neural Networks

Tomasi et al. [42] introduced **Non-local Neural Networks (NLNN)**, intending to capture long-range dependencies. Inspired by non-local mean operation [43], the non-local operation calculates the response at a position computing the weighted sum of the features at all positions, hence the NLNN can be considered as a combination of different "self-attention"-style methods [30]. Buades et al. [43] defined the non-local operation as:

$$h_i' = \frac{1}{C(h)} \sum_{\forall j} f(h_i, h_j) g(h_j) \tag{3.20}$$

In the above equation $f(h_i, h_j)$ generates a scalar between $i$, the index of an output position, and $j$, all possible positions, which represents the relation between them. $g(h_j)$ transforms the input $h_j$, while $\frac{1}{C(h)}$ applies normalization. As discussed above for the MPNL, the two functions $f$ and $g$ can be defined with different settings, resulting to a variety of different models. For the latter, [42] for simplicity reasons only considered using a simple linear transformation, $g(h_j) = W_g h_j$ , while for the former, function $f$, proposed a set of various functions:

1. **Gaussian**: The Gaussian function was proposed as a natural choice of $f$, following the non-local mean [43] and bilateral filters [44]. Considering $h_i^T h_j$ as the dot-product similarity:

$$f(h_i, h_j) = e^{h_i^T h_j} \tag{3.21}$$

$$C(h) = \sum_{\forall j} f(h_i, h_j) \tag{3.22}$$

2. **Embedded Gaussian**: Extending the Gaussian function, the similarity is now calculated in an embedding space:

$$\theta(h_i) = W_\theta h_i \tag{3.23}$$

$$\phi(h_j) = W_\phi h_j \tag{3.24}$$

$$f(h_i, h_j) = e^{\theta(h_i)^T \phi(h_j)} \tag{3.25}$$

$$C(h) = \sum_{\forall j} f(h_i, h_j) \tag{3.26}$$

A noteworthy remark at this point is that the self-attention module [49] introduced for machine translation is a special case of non-local operations in the embedded Gaussian version.

Specifically, for a given $i$, $\frac{1}{C(h)}f(h_i, h_j)$ becomes the softmax computation along the dimension j. As a result $h'$ is calculated as:

$$h' = softmax(h^T W_\theta^T W_\phi h)g(h) \tag{3.27}$$

3. **Dot product**:The function f can also be defined as the dot-product similarity:

$$f(h_i, h_j) = \theta(h_i)^T \phi(h_j) \tag{3.28}$$

$$C(h) = N \tag{3.29}$$

with N being the number of positions in h.

4. **Concatenation**: The function f can also be implemented as using concatenation:

$$f(h_i, h_j) = ReLU(w_f^T [\theta(h_i) \parallel \phi(h_j)]) \tag{3.30}$$

$$C(h) = N \tag{3.31}$$

**Graph Networks**

Battaglia et al. [3] introduced Graph Networks, a more abstract framework that generalizes the above approaches. Specifically, the authors defined the graph as 3-tuple $G = (u, H, E)$, with $u$ being a global attribute, $H = \{h_i\}_{i=1:N^v}$ being a set of $N^v$ nodes and $h_i$ the attributes of i-th node, $E = \{(e_k, r_k, s_k)\}_{k=1:N^e}$ being a set of $N^e$ edges and $e_k$ the edge attributes, $r_k$ the index of the receiver node and $s_k$ the index of the sender node. A Graph Network is described by the **GN block**, the **Computation Steps** and the **Design Principles** described below:

**GN block**. A GN block is consisted of three *update* functions, denoted as $\phi$ and three *aggregation* functions, denoted as $\rho$:

$$e'_k = \phi^e(e_k, h_{r_k}, h_{s_k}, u) \tag{3.32}$$

$$h'_i = \phi^h(\bar{e}'_i, h_i, u) \tag{3.33}$$

$$u' = \phi^u(\bar{e}', \bar{h}', u) \tag{3.34}$$

$$\bar{e}'_i = \rho^{e\to h}(E'_i) \tag{3.35}$$

$$\bar{e}' = \rho^{e\to u}(E') \tag{3.36}$$

$$\bar{h}' = \rho^{h\to u}(H') \tag{3.37}$$

where $E'_i = \{(e'_k, r_k, s_k)\}_{r_k=i,k=1:N^e}$, $H' = \{h'_i\}_{i=1:N^v}$ and $E' = \cup_i E'_i = \{(e'_k, r_k, s_k)\}_{k=1:N^e}$

Functions $\phi^e$ and $\phi^v$ compute per-edge and per-node updates respectively, while $\phi^v$ is applied only once as the global update. The $\rho$ functions receive a set as an input and return the aggregated information as a single element, satisfying the constraints of input permutation invariant and variable numbers of arguments.

**Computation steps**. The computation steps taking place in a GN block are described below:

1. Apply $\phi^e$ on the edges, providing $(e_k, h_{r_k}, h_{s_k}, u)$, to obtain $e'_k$. As defined above, the set of resulting per-edge outputs for each node will be $E'_i$, while the set of all per-edge outputs will be the union of the above sets $E' = \cup_i E'_i$.

2. Apply $\rho^{e\to h}$ to $E'_i$, to aggregate the edge updates for edges that project to vertex i represented by $\bar{e}'_i$.

3. Apply $\phi^h$ to each node $i$, to extract the updated node attribute $h'_i$ and obtain the set of the resulting per-node outputs $H'$, defined above.

4. Apply $\rho^{e \rightarrow u}$ to $E'$, to aggregate all edge updates into $\bar{e}'$.
5. Apply $\rho^{h \rightarrow u}$ to $H'$, to aggregate all node updates into $\bar{h}'$, to be used in the global update.
6. Apply $\phi^{u}$ once per graph and compute the global updated attribute $u'$.

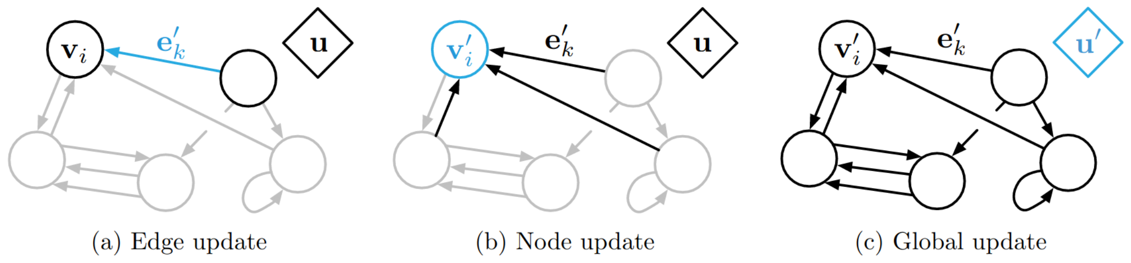


**Figure 3.3.** *Updates in a GN block. Blue indicates the element that is being updated, and black indicates other elements which are involved in the update. Source: [3]*

**Design Principles**. The design of the Graph Network framework is based on satisfying the below three basic principles:

- **Flexible representations**: The GN framework is able not only to support flexible representations of the attributes, but also handle different graph structures. Specifically, the node, edge and global attributes can have arbitrary representational formats, while the output of the GN block can be appropriately modified to satisfy the demands of a specific task. As of the graph structures, the GN framework can by used to structural scenarios where the graph structure is explicitly defined as well as to non-structural scenarios, where the model is expected to infer or assume the relational structure.
- **Configurable within-block structure**: All the functions taking place within the GN block, as well as their inputs can have different settings so that the GN framework can support flexibility in within-block structure configuration.
- **Composable multi-block architectures**: The GN blocks can be considered as "units" to be composed resulting to more complex architectures, using a various number of GN blocks with shared or unshared variables.

## 3.3.3 Variants of Graph Neural Networks

The limitations described in section 3.2.1 have been tackled by a lot of variants of Graph Neural Networks, while research through the years has paved the road not only for even better node and graph representation, but also for handing different types of graphs. These variants can be divided into several categories, which are described in the following subsections:

### Propagation Step

Although all different GNN variants are equipped with a propagation and output step, both of them are of crucial importance in extracting the nodes' and/or edges' hidden states, with the former varying significantly between the variants and the latter being usually a feed-forward neural network. In other words, as described below, the variants employ different ways of aggregating the information of the node's neighbourhood and methods to update nodes' hidden states. The major Graph Neural Networks variants are described below:

- **Convolution**: These methods are divided in two different approaches.

  **Spectral approaches**, which utilize a spectral representation of the graphs. These approaches, however, are inadequate to handle graphs with a different structure than the one used in training, since the learned filters depend on the Laplacian eigenbasis, which depends on the graph structure. The most important ones are:

  – **Spectral Network**[45]: applies convolution in the Fourier domain by computing the eigendecomposition of the graph Laplacian. For a signal $x \epsilon \mathbb{R}^N$ with a filter $g_\theta = diag(\theta)$ parameterized by $\theta \epsilon \mathbb{R}^N$: $g_\theta \circledast x = U_{g_\theta}(\Lambda) U^T x$

  – **ChebNet**[46], which utilizes a K-localized convolution to define a convolutional neural network which could remove the need to compute the eigenvectors of the Laplacian.

  – **GCN**[14], which by limiting the layer-wise convolution operation to K = 1 to alleviate the problem of overfitting on local neighborhood structures for graphs with very wide node degree distributions by applying a set of approximations, generalized the definition to a signal $X \epsilon \mathbb{R}^{NxC}$ with $C$ input channels and F filters for feature maps as:

  $$Z = \widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}} X \Theta \tag{3.38}$$

  This model proved itself very efficient on a wide variety of tasks and paved the road for a lot of GNN models that followed.

  **Non-spectral approaches**, which apply convolutions directly on the graph, aggregating information from each node's neighbours. Handling neighborhoods of various sizes and maintaining the local invariance of CNNs, below are the most successful models:

  – **Neural FP** [47], applying different weights for nodes with different degrees:

  $$x = h_v^{t-1} + \sum_{i=1}^{|N_u|} h_i^{t-1} \tag{3.39}$$

  $$h_v^t = \sigma(x W_t^{|N_u|}) \tag{3.40}$$

  where $W_t^{|N_u|}$ is the weight that corresponds to nodes with degree $|N_v|$

  – **Diffusion-Convolutional Neural Networks (DCNNs)**[48], where transition matrices are used to define the neighborhood of the nodes. With **X** being the input features of shape $N \times F$, where N is the number of nodes and F the number of features, **P** being the degree-normalized transition matrix from the graph's adjacency matrix and **P*** being an $(N \times K \times N)$ tensor that contains the power series $P, P^2, ..., P^K$ of matrix P, the defusion representation **H** of each node is defined as:

  $$H = f(W^c \odot P * X) \tag{3.41}$$

  The graph embedding, usually used for graph classification, can be acquired as:

  $$H = f(Wc \odot 1_N^T PX/N) \tag{3.42}$$

  – **Dual Graph Convolutional Network (DGCN)**[49] employs two convolutional networks to capture the both local and global consistency and then uses an unsupervised loss to ensemble them. The first convolutional network is as proposed in GCN[14], while the second one utilizes a positive pointwise mutual information (PPMI) matrix instead of the adjacency matrix. With $X_P$ being the PPMI matrix and $D_P$ the diagonal degree matrix of $X_P$:

$$H' = \rho(D_{\rho}^{\frac{-1}{2}} X_{\rho} D_{\rho}^{\frac{-1}{2}} H\Theta) \tag{3.43}$$

– **GraphSAGE** [50] computes the embeddings by uniformly sampling a certain number of neighbours and then aggregating their features, through the below equations:

$$h_{N_v}^t = AGGREGATE(h_u^{t-1}, \forall u \epsilon N_v) \tag{3.44}$$

$$h_v^t = \sigma(W^t[h_v^{t-1} \parallel h_{N_v}^t]) \tag{3.45}$$

Hamilton et al. [50] Suggested three different aggregation methods:

1. **Mean aggregator**:

$$h_v^t = \sigma(W * MEAN(h_v^{t-1} \cup h_u^{t-1}, \forall u \epsilon N_v) \tag{3.46}$$

The mean aggregator is also the only one that doesn't apply the concatenation in the general equation and hence it could be viewed as a "skip connection" [51]

2. **LSTM aggregator**: By using an LSTM-based aggregator, a better expressive capability is achieved, but in order to satisfy the permutation invariant constraint, [50] permutes each node's neighbours to be able to apply the LSTM on an unordered set.

3. **Pooling aggregator**: Neighbours' hidden states are fed through a fully connected layer and then a max-pooling operation is applied:

$$h_{N_v}^t = max(\sigma(W_{pool} h_u^{t-1} + b), \forall u \epsilon N_v) \tag{3.47}$$

- **Gate**: In order to enhance the long-term propagation of information across the graph, there have been attempts to employ gate mechanisms in the propagation step. Specifically, [52] proposed the **Gated Graph Neural Network (GGNN)**, where Gated Recurrent Unit is employed in the propagation step applyin the recurrence for $T$ steps. In the below equations, after aggregating information from its neighbours, present in $A_u$, node $u$ updates its hidden state with a GRU-like update function that operates on the aggregated information and the previous hidden state. **a** aggregates the neighbours' messages, **z** is the update and **r** is the reset gate:

$$a_v^t = A_v^T[h_1^{t-1} \cdots h_N^{t-1}]^T + b \tag{3.48}$$

$$z_v^t = \sigma(W^z a_v^t + U^z h_v^{t-1}) \tag{3.49}$$

$$r_v^t = \sigma(W^r a_v^t + U^r h_v^{t-1}) \tag{3.50}$$

$$\hat{h_v^t} = tanh(W a_v^t + U(r_v^t \odot h_v^{t-1})) \tag{3.51}$$

$$h_v^t = (1 - z_v^t) \odot h_v^{t-1} + z_v^t \odot \hat{h_v^t} \tag{3.52}$$

- **Attention**: [4] proposed **Graph Attention Network**, which adopts a *self-attention* strategy in the propagation step, generating the hidden states of each node by attending its neighbours. Specifically, an arbitrary graph attention network is constructed by stacking building block layer, defined as shown below:

$$a_{ij} = \frac{exp(LeakyReLU(a^T[Wh_i \parallel Wh_j]))}{\sum_{k\epsilon N_i} exp(LeakyReLU(a^T[Wh_i \parallel Wh_k]))} \tag{3.53}$$

$$h_i' = \sigma(\sum_{j\epsilon N_i} a_{ij} W h_j) \tag{3.54}$$

$a_{ij}$ is the attention coefficient of node j to node i, while $h_i$ is the output features of each node.

In addition, [4] also proposed the use of *multi-head attention*, where K independent attention mechanisms are applied to compute the hidden states. The final output features can then be extracted by either concatenating the K different features or by employing *averaging*, if we perform multi-head attention on the final layer of the network, as shown respectively below:

$$h_i' = \parallel_{k=1}^{K} \sigma(\sum_{j\epsilon N_i} a_{ij}^k W^k h_j) \tag{3.55}$$

$$h_i' = \sigma(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\epsilon N_i} a_{ij}^k W^k h_j) \tag{3.56}$$

where $a_{ij}^k$ is the normalized attention coefficient computed by the k-th attention mechanism.

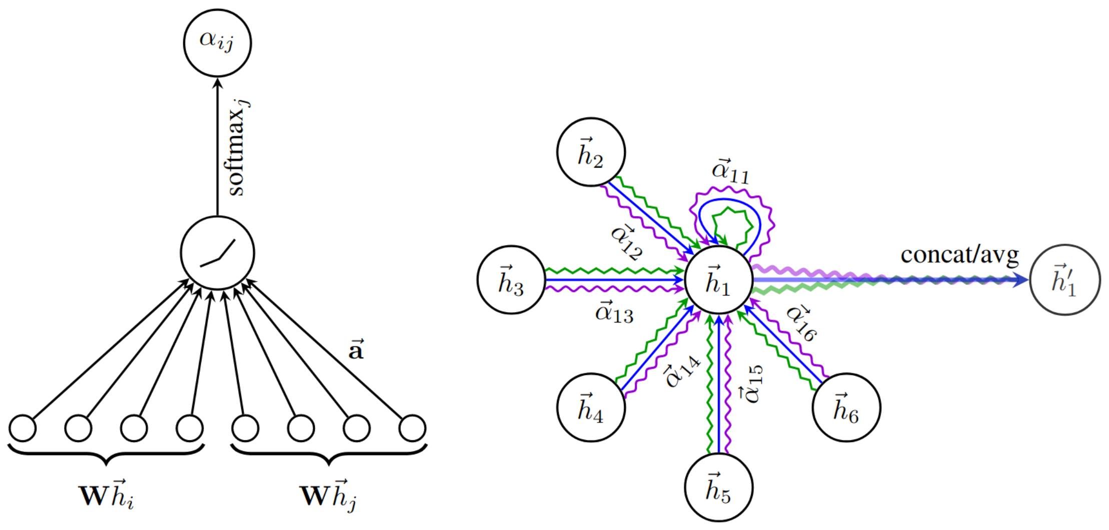


**Figure 3.4.** ***Left***: *The attention mechanism $a(W\vec{h}_i, W\vec{h}_j)$ employed by GAT, parametrized by a weight vector $\vec{a}\epsilon\mathbb{R}^{2F'}$, applying a LeakyReLU activation.* ***Right***: *An illustration of multihead attention (with K = 3 heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain h1. Source: [4]*

### Types of Graphs

Unlike the graphs that can be considered as input for the original GNN, which are comprised of informative labeled nodes and undirected edges, there are also many other types of graphs, which can be grouped in the following categories:

- **Directed Graphs**: Directed edges may carry more valuable information than undirected. The most representative example of such graphs are the knowledge graphs. [53], in order to address the problem of handling directed graphs, suggested two types of weight matrix $\mathbf{W_p}$ and $\mathbf{W_c}$. The propagation rule is shown as follows:

$$\mathbf{H^t} = \sigma(\mathbf{D_p^{-1}A_p}\sigma(\mathbf{D_c^{-1}A_cH^{t-1}W_c})\mathbf{W_p}), \tag{3.57}$$

where $D_p^{-1}A_p, D_c^{-1}A_c$ c are the normalized adjacency matrix for parents and children respectively.

- **Heterogenous Graphs** In an Heterogenous Graph there exist different types of nodes. While a straightforward way to handle these graphs is converting the type of each node to a one-hot feature vector and concatenating it with the original feature, GraphInception [54] proposed grouping the neighbors according to their node types and distances. Each neighbourhood group is then treated as sub-graph in an homogeneous graph and then all propagation results from the homogeneous graphs are concatenated. Lastly, heterogeneous graph attention network (HAN) was introduced by [55] which utilizes node level and semantic-level attentions.

- **Graphs with Edge Information** To properly encode graphs with valuable edge information, [55] proposed converting the graph to a bipartite graph where the original edges also become nodes and one original edge is split into two new edges which means there are two new edges between the edge node and begin/end nodes. Another solution was introduced with r-GCN [5], which adapts different weight matrices for the propagation on different kinds of edges and introduces two kinds of regularization to reduce the number of parameters for modeling amounts of relations: basis- and block-diagonal-decomposition. This model or variants of it is being usually applied to Knowledge Graphs, which are graphs with multiple types of relations, which carry significant information. It is also being used in some of our experiments in this work.

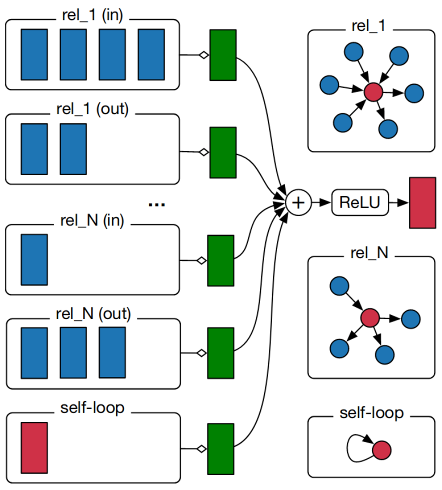


**Figure 3.5.** *Diagram for computing the update of a single graph node/entity (red) in the R-GCN model. Activations (d-dimensional vectors) from neighboring nodes (dark blue) are gathered and then transformed for each relation type individually (for both in- and outgoing edges). The resulting representation (green) is accumulated in a (normalized) sum and passed through an activation function. This per-node update can be computed in parallel with shared parameters across the whole graph. Source: [5]*

- **Dynamic Graph**, which has static graph structure and dynamic input signals. [56] and [57] managed to capture both kinds of information, by firstly collecting spatial information utilizing GNNs, then feeding the outputs into a sequence model like sequence-to-sequence model or CNNs. Both models were employed for the task of traffic prediction on graph roads. On the other hand, [58] and [59] followed a different approach, by collecting spatial and temporal messages simultaneously. These models extend static graph structure with temporal connections so they can apply traditional GNNs on the extended graphs.

## 3.3.4 How to use GNNs?

After applying a number $k$ of GNN layers, the latent $h_i^k$ features for each node $i$ are generated. These features can then be used for a variety of different tasks. Some of the most common practices is to use the final node states for:

- **Node Classification**, to classify the nodes in a set of categories
- **Graph Classification**, to classify the whole graph, by aggregating all the latent node features of the graph to extract a graph embedding
- **Link Prediction**, in order to infer the existence or the type of an edge between a pair of nodes.

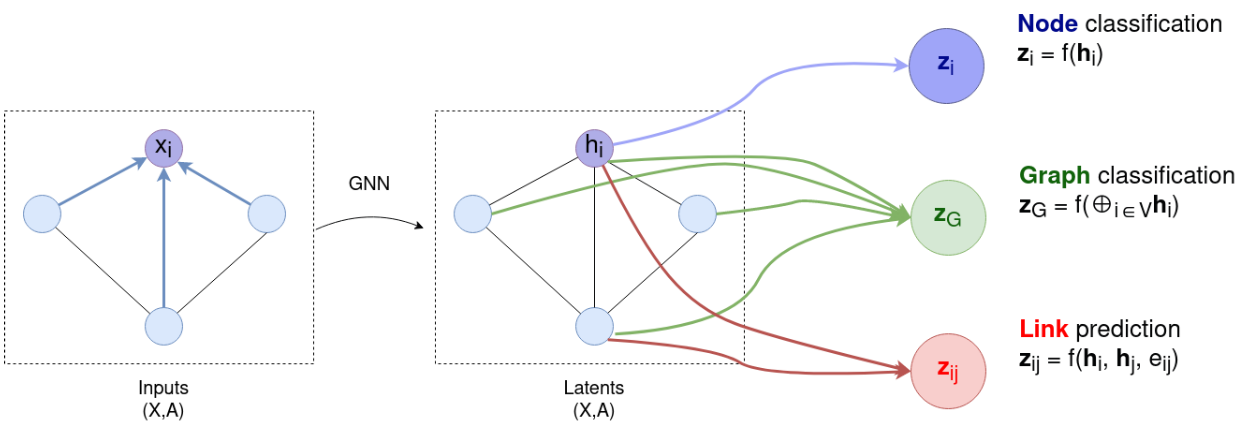


**Figure 3.6.** *Common use cases of GNNs*

All the above methods have various applications in fields not limited to social science (social networks) [13], [14], natural science (physical systems [15], [16] and protein-protein interaction networks [17]).

## 3.3.5 Knowledge Graphs - ConceptNet

Machine learning on tasks relevant to language can be significantly boosted by enriching it with specific knowledge and sources of external information. The concept of knowledge graph was first introduced by Google in 2012. It is defined as a large-scale knowledge base composed of a large number of entities and relationships between them. Through the years, knowledge graph, used as a semantic network, has drawn a lot of attention and has been proved effective in natural language processing and tasks such as intelligent question answering system, intelligent recommendation system and so on. Combined with big data and deep learning, knowledge graphs now have become one of the core driving power for the development of artificial intelligence.

One popular general knowledge graph is ConceptNet, proposed by [60]. ConceptNet utilizes labeled edges to connect words and phrases of natural language. The information within ConceptNet was gathered from sources such as include crowd-sourcing, games with a purpose and expert-created resources. The motivation behind it was to represent the general knowledge involved in language,

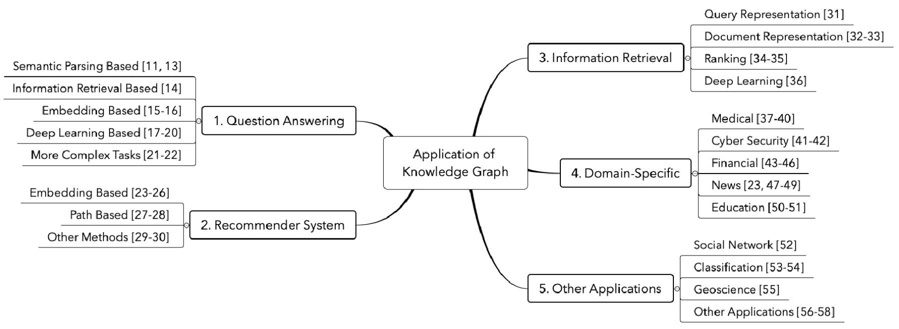


**Figure 3.7.** *Application fields of Knowledge Graphs. Source: [6]*

in order to enable models to better comprehend the meanings behind the words used in natural language by humans and to pave the road not only to better performances on various tasks, but also to new exciting tasks that require external knowledge and would otherwise standard models would perform poorly.

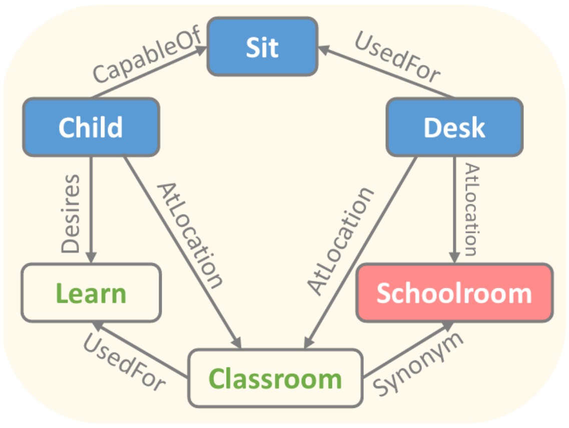


**Figure 3.8.** *An example sub-graph of a Knowledge Graph. Source: [7]*

The construction of ConceptNet, a Multilingual Graph, was completed gathering information from the below sources:

- Facts acquired from Open Mind Common Sense (OMCS) [61] and sister projects in other languages
- Information extracted from parsing Wiktionary, in multiple languages, with a custom parser ("Wikiparsec")
- "Games with a purpose" designed to collect common knowledge [62]
- Open Multilingual WordNet [63], a linked-data representation of WordNet [64] and its parallel projects in multiple languages
- JMDict [65], a Japanese-multilingual dictionary
- OpenCyc, a hierarchy of hypernyms provided by Cyc, a system that represents common sense knowledge in predicate logic
- A subset of DBPedia [66], a network of facts extracted from Wikipedia infoboxes.

ConceptNet employs a defined set of relations (e.g. *IsA, UsedFor, CapableOf*) in order the representation of the relationships to be independent of the language or the source of the terms it connects. The knowledge graph is as a result a directed graph, while it also has a few relations designated as symmetric, such as *SimilarTo*. As also observed by [60], relations with specific semantics, such as *UsedFor* and *HasPrerequisite*, tend to connect common words and phrases, while rarer words are connected by more general relations such as *Synonym* and *RelatedTo*. An analysis on the frequency of each relation in the extracted subgraphs of ConceptNet for our task will be presented in the next sections.

Chapter 4

# Introduction to NLP and Visual Dialog

## 4.1 What is Visual Dialog?

The task of Visual Dialog, as proposed by [11], requires an AI agent to hold a meaningful dialog with humans in natural, conversational language about visual content. Given as an input an image, a dialog history and a question referring to the image, the desired model is expected to combine the question with the scene shown in the image, extract any useful information from the dialog history and come up with the an accurate response.

As a lot of research has been conducted towards the direction of language-only dialog, achieving remarkable results (e.g. Alexa, Siri, etc), the ability to engage in meaningful conversation about visual content in natural language with people will be a requirement for the next generation of visual intelligence systems. The applications of such systems would be beneficial in various domains, including:

- Aid visually impaired people towards understanding their surrounding environment, or a given situation shown in an image, which fits exactly to the task of Visual Dialog
- Achieve a more useful, interesting and engaging interaction with AI assistants, as the user could ask questions about a situation the device has access to.

Furthermore, as current works focusing on language-only dialog distinguish two types interaction: **goal-driven dialog** and **goal-free dialog**, Visual Dialog resembles a lot the the latter category, while it can also be proven very helpful for the former category. Specifically, goal-driven dialog is evaluated on task-completion rate or on the time needed to achieve the completion of the task. For example, a user could use a bot to get informed about the availability and book a flight ticket. Similarly, in Visual Dialog the user can use the agent to retrieve all the information they need for a specific task. In the other hand, goal-free dialogues resemble to casual conversations with chat-bots, hence the longer the user engagement and interaction, the better. Visual Dialog is a good approach for this kind of conversations referring to visual content. As stated by [11], it is disentangled enough from a specific downstream task so as to serve as a general test of machine intelligence, while being grounded enough in vision to allow objective evaluation of individual responses and benchmark progress. The former discourages task engineered bots for ‘slot filling’ [67] and the latter discourages bots that put on a personality to avoid answering questions while keeping the user engaged.

Consider the Visual Dialog example shown in Fig 4.1. The first question *'Is motorcycle moving or still?'* requires the machine to selectively focus and direct attention to a relevant region of the image where the motorcycle is shown. The same stands for the next question *'What kind of dog is it?'*, while the model is also required to understand the the *dog* refers to the one present in the caption and the image. The third question *'What kind of dog is it?'* requires co-reference

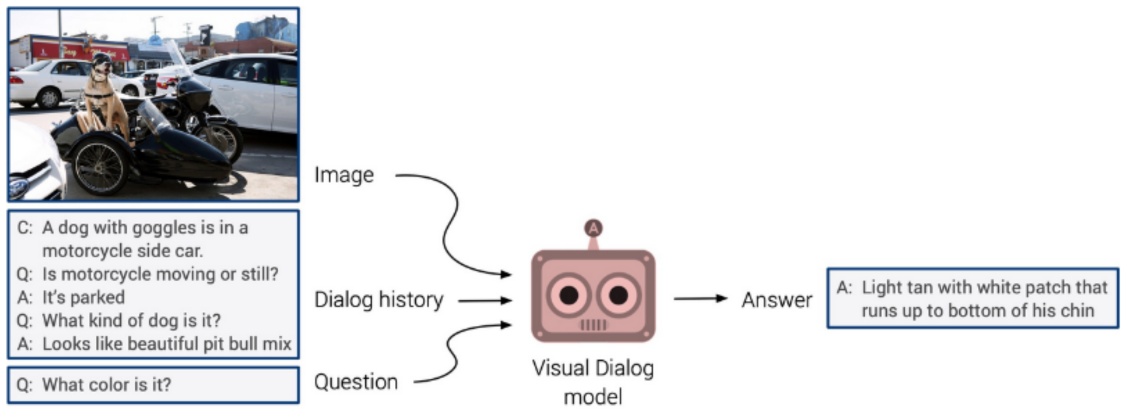

C: A dog with goggles is in a motorcycle side car.
Q: Is motorcycle moving or still?
A: It's parked
Q: What kind of dog is it?
A: Looks like beautiful pit bull mix
Q: What color is it?
Image
Dialog history
Question
Visual Dialog model
Answer
A: Light tan with white patch that runs up to bottom of his chin

**Figure 4.1.** *An example dialog of the Visual Dialog task. Source: visualdialog.org*

resolution (whom does the pronoun 'it' refer to?) and the model should be able to infer which object the question refers to by examining the dialog history.

In order a trained model to address effectively the task and respond with meaningful and accurate responses, it should to be able to recognize the image's main objects and the environment they are in, infer context from history to capture any relation between the current question and a previous one, combine the two sources of information and draw a conclusion to generate a response. As a result an effective model architecture should be equipped with **Natural Language Processing** and **Computer Vision** methods and utilize **Multimodal Fusion** techniques to combine the two modalities (Vision and Text). In the next sections we present the basic concepts of the above fields and focus on the methods that are being used in the current work.

## 4.2 Natural Language Processing

The field of Natural Language Processing (NLP) covers a wide range of topics, which involve computational processing and understanding of human language. It has gained a lot of research interest through the last years, as its applications vary from speech recognition to extracting information and question answering and are becoming an increasing part of every day's life. Although the field initially relied mostly on statistics and probability [68], today's artificial neural networks assembling complex and vast deep learning models have paved the road for incredible results in every Natural Language Processing task. Moreover, the training of such deep architectures was made possible by the abundance of data in massive datasets, constructed by sophisticated data collection procedures.

The core areas of Natural Language Processing tackle underlying problems such as:

- Language Modeling, which underscores quantifying associations among naturally occurring words
- Morphological Processing, which handles segmentation of meaningful components of words and identifies the true parts of speech of words as used
- Syntactic Processing, which generates sentence diagrams as possible precursors to semantic processing
- Semantic Processing, which attempts to distill meaning of words, phrases, and higher level components in text.

### 4.2.1 Applications

Natural Language Processing and its applications have been present in our lives for the last few years. From changing the way we interact with most of our daily used devices, to shaping flexible and complex search engines and to giving life to personalized voice assistants, NLP has affected more aspects of our lives than most people realise. The applications of Natural Language Processing vary a lot and a few representatives are presented below:

- **Information extraction**: The automated retrieval of relevant to a selected topic information from a text or a set of texts. More formally, IE is the process of extracting structured information from unstructured and/or semi-structured documents and other machine-readable sources of information.

- **Information Retrieval**: The task of searching for documents or information and metadata within them, as well as searching databases and the World Wide Web.

- **Machine Translation**: The process creating a machine capable of translating text between and among natural languanges languages.

- **Text Summarization**: The technique for generating a brief and accurate summary of vast texts, preserving the general meaning as well as locating the most important parts and distilling the essential information.

- **Document Classification**: The task of assigning a document to one or more categories.

- **Document Clustering**: The task of cluster analysis to textual documents, e.g. dividing a set of document collections into different number of groups based on Document contents-similarity

- **Question Answering**: The process of automatically generating answers to questions posed by humans in natural language. The desired model has to be able to comprehend the question and answer properly.

- **Text Generation**: The task of receiving keywords or a sentence as input and generating text indistinguishable to human-written text. Interesting use cases are Code generation and Stories generation.

- **Speech Recognition and Synthesis**: the task of extracting textual representations of a spoken utterance.

### 4.2.2 Earlier Approaches

As Natural Language Processing is shaped today, it is mainly a data-driven field using statistical and probabilistic computations along with machine learning. In the past however, machine learning methods such as naive Bayes, k-nearest neighbors, hidden Markov models, conditional random fields, decision trees, random forests, and support vector machines were the dominant approaches. However, during the past several years, there has been a wholesale transformation, and these approaches have been entirely replaced, or at least enhanced, by neural models, as discussed in the next sections.

### 4.2.3 Word Embeddings

In Natural Language Processing, word embedding stands for the way words are represented. The input of NLP models can not be the words as in the given natural language, rather numbers that represent the words' meaning. The ideal word representation would include the word meaning, or the concept that this word expresses, as well as some attributes such as the concept of word similarity and difference, and the ability to distinguish between polysemous words, that have various meanings depending on the context they are used. These word representations have been proven to be crucial in order to achieve further achievements in the field of NLP.

Research through the years has been made towards two different approaches, when representing a word. On the one hand *Denotational Representation* suggests representing a word as a symbol, while on the other hand *Distributional Semantics* is the method of representing a word based on the context in which it usually appears, following the Distributional hypothesis, which states that linguistic items with similar distributions have similar meanings.

### 4.2.4 Denotational Representation

This method of representing a given vocabulary's words suggests assigning a value to each word. This value can either be a scalar value or a vector.

The simplest approach of mapping all words of a vocabulary $V$ to a symbol is by using **Vocabulary IDs**. Specifically, each word is assigned to a unique word ID, which is usually an increasing integer value. Given a vocabulary $V = \{w_1, w_2, ..., w_{|V|}\}$ each word $w_i$ is assigned to the integer $i$:

$$e^{w_1} = 1, e^{w_2} = 2, \ldots, e^{w_{|V|}} = |N| \tag{4.1}$$

As it is shown above, this approach requires the use of $|V|$ different word IDs, which lead to computational inefficiency. More importantly, this approach does not effectively represent the word's meaning and is also not able to capture the similarities and difference between words as well as deal with the ambiguity problem.

A more popular approach is the use of **One-Hot-Encoding**. This method suggests the use of $\mathbb{R}^{|V|x1}$ vectors to represent each word. By increasing the dimensionality further improvements can be accomplished using compression techniques. For a vocabulary $V = \{w_1, w_2, ..., w_{|V|}\}$ the One-Hot-Encoding of its words would be:

$$e^{w_1} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad e^{w_2} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \quad , \ldots, e^{w_{|V|}} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{4.2}$$

This method, however, also fails to capture the similarities and differences between words, while the ambiguity problem also remains unsolved.

### 4.2.5 Distributional Semantics

Approaches that follow the Distributional hypothesis, generating word embeddings in a way that words with similar meanings are represented in a similar way, were proved to be much more efficient, tackling all the above problems. Denotational Semantics word vectors can be categorized as sparse or as dense word vectors.

### Sparse word vectors

While there have been introduced several Sparse word vectors, we present one of the most popular ones:

**Term Frequency–Inverse Document Frequency (TF-IDF) Representation**: It is a numerical statistic method that is utilized to calculate the importance of a word for a document in a collection or corpus. Usually used as a weighting factor, the tf–idf value increases proportionally to the number of times a word appears in the document and is restricted by the number of documents in the corpus that contain the word, which contributes to capture the fact that some words appear more frequently in general.

The td-idf values utilizes two terms: The term frequency (tf) and the inverse document frequency (idf). The TF term is used to express that words that occur more frequently are usually more important than the less frequent ones. The term IDF, however, aims to shape the weights so as to follow the intuition that too frequent words (determiners, pronouns etc) are more likely to be less important. Hence, the tf-idf product tries to find the trade-off between the tf and the idf term.

### Dense word vectors

In many use cases sparse word vectors are not selected for representing words, due to their high dimensionality, as they are very long vectors, with a size of approximately around 50,000. This characteristic renders sparse representations computationally expensive. In contrast, dense word vectors are much sorter with a dimensionality, around 300-1200. This difference makes them much more efficient, as now the classifiers have to learn much fewer weights. Moreover, models trained using dense word vectors tend to perform better than using sparse vectors. This phenomenon could be possibly explained by the fact that lower dimensionality helps boost generalization and avoid overfitting. In addition, the dense representations generally achieve a better understanding of the meaning of the word.

While a lot of research has been conducted towards generating dense word representations, with notable approaches utilizing feedforward neural networks ([69]) or Recurrent Neural Networks, significant breakthrough was made when Mikolov proposed the famous Word2Vev [8]. Word2Vec is a shallow, two-layer neural network that is trained to reconstruct linguistic contexts of words. Given an input of a large corpus of words, it produces a vector space, typically of some hundred dimensions, with each word in the corpus being assigned a corresponding vector in the space. An important property of that space is that words that share similar contexts in the corpus are located close to one another in that space. Word2Vec has two forms, the **Continuous Bag Of Words (CBOW)** model and the **Skip-Gram** model as depicted in Fig. **??**. In the CBOW model the distributed representations of context are combined to predict the word in the middle. All words get projected into the same position, by averaging their vectors and as a result the order of words does not influence the projection. Hence, the name Bag Of Words.In the Skip-gram model, the distributed representation of the input word is used to predict the context.

GloVe Embeddings One of the most popular word embeddings today that has been proved very efficient in Natural Language Processing is GloVe [70]. It is an unsupervised learning algorithm and the training is performed on aggregated global word-word co-occurrence statistics from a corpus, while the resulting representations have interesting linear substructures of the word vector space. The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence. The central intuition behind the model is the phenomenon that ratios of word-word co-occurrence probabilities can encode some form of meaning. The GloVe model is trained on a global word-word co-occurrence matrix, which demonstrates how

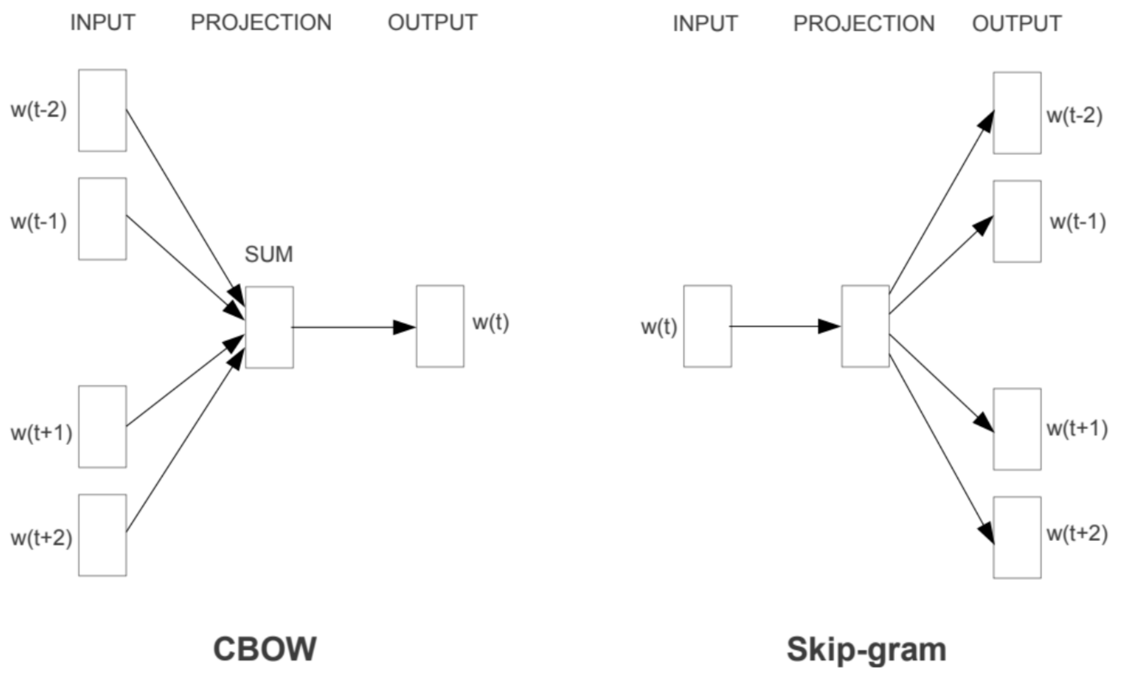


**Figure 4.2.** *The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. Source: [8]*

frequently words co-occur with one another in a given corpus.

### 4.2.6 Embeddings from Language Models (ELMo)

ELMo embeddings [71] are deep contextualized word representations that offer high-quality representations for language, modeling both complex characteristics of word use and adjusting them in different linguistic contexts. The vectors are generated by a bi-directional LSTM trained with a coupled language model (LM) objective on a large text corpus. ELMo representations are a function of all the internal layers of the bi-directional language model. Lastly, one drawback of ELMo is the need for tunning for every different task.

## 4.3 Computer Vision

Computer vision is the field, which works to help computers gain high-level understanding from digital images or videos. Tasks of this field include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, while some popular sub-domains of computer vision include scene reconstruction, object/event detection, object recognition and 3D pose estimation.

In general, in computer vision the image data can take many forms, from simple images to video sequences, views from multiple cameras, multi-dimensional data from a 3D scanner and medical scanning device. In our task, however, the image data is limited to a single image, as it is detailed described in Section 5.2.

Images are usually represented as 3-dimensional volumes $HWC$, with $H$ being the image height, $W$ the width and $C$ the number of channels. Having three channels for defining the red, green, and blue color components of each individual pixel, RBG images are described with very high dimensional representations. As a result, feeding to a fully-connected network an image of typical size 224×224. leads to a number of weights for a single neuron in the input layer equal to 224  224

3 = 150.528. Hence, sparse connectivity and parameter sharing are the main advantages of CNNs the render them ideal for Computer Vision tasks.

Convolutional Neural Networks were briefly presented in Section 2.4.2, but here we will focus on CNN architectures for object detection. These architectures have a few differences compared to original CNNs, that enable them to model the localization of the objects in the image. Specifically, instead of producing only scores for candidate labels, an object detection system must output a tuple for each detected object: the class scores and the bounding box coordinates. Since each image contains an arbitrary number of objects, most models determine a group of region proposals and then use a classifier to compute the probability of an object being present in that region. The regions proposals are generated by applying multiple anchor boxes at different positions of the image, which are bounding boxes of various shapes and sizes.

## 4.3.1 Faster R-CNN

One of the most efficient models for object detection that has met an increasing popularity over the years is the Faster R-CNN [9], depicted in Fig. 4.3:

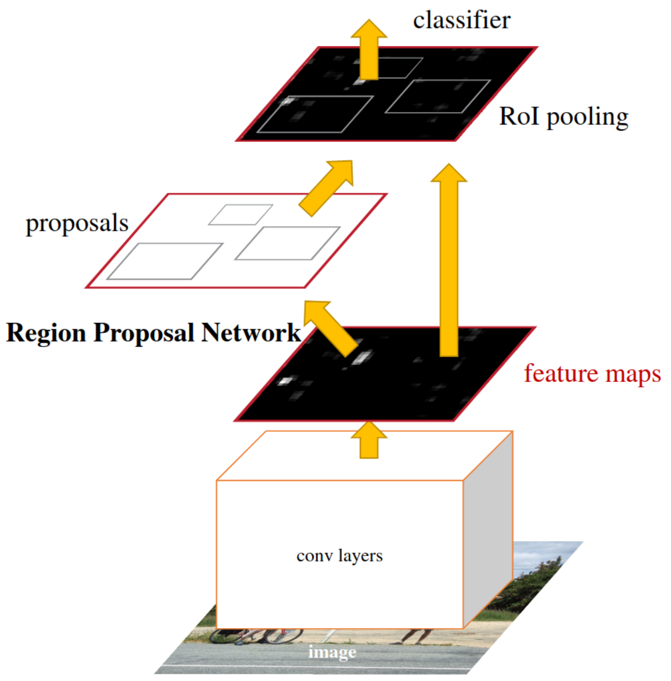


**Figure 4.3.** *An overview of the Faster R-CNN architecture. The RPN module serves as the 'attention' of the unified network. Source: [9]*

In a Faster R-CNN model the input image is firstly passed through a set of convolutional layers to obtain the feature map of the image. On this feature map a [9] proposed to apply a *Regional Proposal Network (RPN)*, which utilizes the features of each anchor box to compute to probability of an object being present in the region specified by the anchor, which is often called as "objectness" score. Since the regions proposed by the RPN have varying sizes, the same stands for the feature maps corresponding to these regions. In order to generate representations of a fixed size, [9] proposed to apply a *Region of Interest (RoI)* pooling layer to the feature maps, which, unlike a Max-Pooling layer, splits the feature map into a fixed number of regions, and keeps the maximum value in every region. As a result, the size of the output is the same regardless of the size of input. Lastly, to obtain the desired output, the features of the proposals are passed to a classifier that outputs class label probabilities and to a linear regression layer that refines the coordinates of the bounding box.

Chapter 5

# Proposed Model

## 5.1 Introduction

Dialog is an efficient way for humans to communicate, exchange information and share emotions. Due to this effectiveness it is an important research goal to develop artificial intelligence based agents capable of conducting conversation with humans. When humans converse, however, subtle details and nuances are often crucial. The significance of recognizing fine details and nuances, is what makes creating agents for visual dialog a challenging endeavor. Research through the years, seeking methods to facilitate human-computer conversation about images, has introduced and made progress on tasks such as image captioning, visual question answering, visual question generation and very recently also visual dialog. Visual Dialog has drawn increasing research interest. In this task, an image is provided as context input, accompanied with a summarizing caption and a dialog history of question-answer pairs. The goal is to answer questions posed in natural language about the image.

## 5.2 Dataset

The task of visual dialog and the corresponding dataset, which is being used for the next experiments was proposed by [72]. The challenge of this task is to train a model to be able to answer follow up questions, given an image, a dialog history, and a question about the image. Specifically, the agent has to combine the current question with the image, infer context from history, and answer the question appropriately. Being decoupled enough from a specific downstream task, Visual Dialog functions as a general test of machine intelligence, while it is also sufficiently anchored in vision allowing impartial evaluation of individual responses and progress benchmarking.

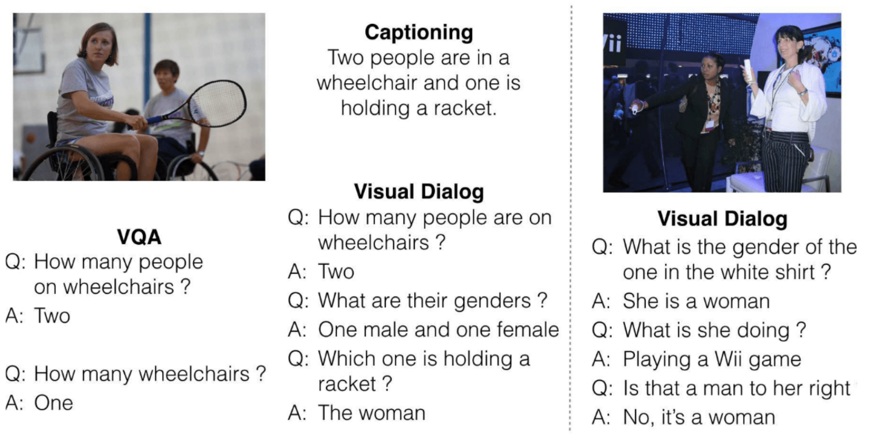
The images corresponding to a different dialog are collected from the Common Objects in Context (COCO) [73] dataset, which depicts various objects in everyday scenes. The visual complexity of these images enables the progress of engaging and diverse conversations. As stated by [73], good data for the Visual Dialog task should include dialogs that have (1) temporal continuity, (2) grounding in the image, and (3) mimic natural 'conversational' exchanges. In order to satisfy these constraints, the dialogs were generated by two paired workers on AMT to chat with each other in real-time. The one worker is assigned to the 'questioner' role and the other to the 'answerer' role. The former has access only to the caption of the image, without seeing the actual COCO image, and their task is to ask questions that will enable them to 'imagine the scene better'. The latter, the 'aswerer' has access to both the image and its caption and their task is to answer questions asked by their chat partner. The workers were encouraged to reply as naturally and 'conversationally' as possible, instead of providing short or concise answers. Each dialog ended after the exchange of a total of 20 messages, which results to 10 pairs of questions and answers.

The VisDial dataset is consisted of 123,287 images, with one dialog of 10 rounds for each image, resulting on total of 1,232,870 QA pairs, on the training split, while the validation split is composed of 2,064 images x 10 rounds and the test split of 8,000 images x 1 round. All the images are collected from the famous MSCOCO [10] dataset. Both training and validation splits have complete dialogues of 10 rounds, while the test split has a random number of answered rounds (between 1 and 9) and one round to be answered by the model.

**Figure 5.1.** *Examples of COCO images [10]*

**Comparing with the VQA dataset** One could compare the visual dialog task, with the popular Visual Question Answering dataset (VQA)[74], which describes the task of providing an image to the model and asking one question. Additional similar datasets are Visual 7W [75] and Baidu mQA [76], but a brief search on publications is enough to conclude that the VQA is much more popular and studied dataset, with numerous scientific publications achieving impressive results. However, the concept of the dialog, having 10 rounds instead of just one and therefore expecting that the model will also take into account the history of the dialog, is not the only significant difference between the VisDial and the above datasets. One key differentiation is that in all previous datasets, subjects saw an image while asking questions about it. As suggested by [77], [78] and [79] this tends to introduce a bias, when generating the dataset. Specifically, a vast amount of questions is questioning the existence of objects that are actually found in the picture. This allows language-only models to perform remarkably well on VQA. Interestingly, for a specific type of questions in the VQA dataset starting with ‘Do you see a . . . ’, blindly answering ‘yes’ without reading the rest of the question or looking at the associated image results in an average VQA accuracy of 87%. In VisDial, questioners have not access to the image and, as a result, this bias is reduced. Finally, another significant difference between Visual Dialog and VQA is that in the former the agent is primed with a caption for the image and questions often refer to objects referenced in it, while in the latter task there is no image caption. As a result, the two tasks are not similar even on the first round of the dialog.

In Fig. 5.2 two examples of the two datasets are shown to demostrate the differences between them. On the left, for the same image two questions from the VQA datasets are shown (in VQA an image may be used in more than one example) and one dialog from the Visual Dialog dataset. As one can easily discern, besides the number of questions per example (only one question vs a dialog of 10 rounds), there are significant differences in the questions themselves. Specifically, while VQA questions are simple and complete, Visual Dialogue’s questions depend on one another, as they carry pronouns referring to objects of previous questions. This nature of the questions not only resembles a real dialog, but also requires the model to be able to infer the object each pronoun refers to. Finally, for the same COCO image the desired result of the task of image captioning is shown, which is the task of annotating images with natural language at the scene level. On the right, another example dialog of the Visual Dialog dataset is depicted.



**Captioning**
Two people are in a wheelchair and one is holding a racket.

**VQA**
Q: How many people on wheelchairs ?
A: Two

Q: How many wheelchairs ?
A: One

**Visual Dialog**
Q: How many people are on wheelchairs ?
A: Two
Q: What are their genders ?
A: One male and one female
Q: Which one is holding a racket ?
A: The woman

**Visual Dialog**
Q: What is the gender of the one in the white shirt ?
A: She is a woman
Q: What is she doing ?
A: Playing a Wii game
Q: Is that a man to her right
A: No, it's a woman

**Figure 5.2.** *Differences between image captioning, Visual Question Answering (VQA) and Visual Dialog [11]*

The common points the Visual Dialog and VQA share, as well as their differences are grouped in the below tables:

| **Common points** |
|---|
| Question over an Image |
| Agent has to combine image and text modalities |
| Images from COCO dataset [10] |

**Table 5.1.** *Common points between Visual Dialog and Visual Question Answering*

| **Different Points** | |
|---|---|
| **Visual Dialog** | **Visual Question Answering** |
| Image and Caption as initial input | Image as input |
| 10 rounds of questions | Only one question |
| Questioners do not see the image | Questioners see the image |

**Table 5.2.** *Differences between Visual Dialog and Visual Question Answering*

One final comment on the differences between Visual Dialog and VQA is the amount of research conducted in these two tasks. Specifically, as noted above, the latter has drawn the interest of many researchers, resulting in a vast amount of publications undertaking the specific task, while it is also a few years older than the former. On the other hand, Visual Dialog dataset has been less popular and there could be more space for insightful future work. A detailed overview of the related work conducted on the Visual Dialog dataset is presented in the next section.

### Problem Formulation

As proposed by [11], in the visual dialogue task, an agent is required to reason based on a given image I , caption C, dialogue history $H_t = \{C, (Q_1, A_1), ..., (Q_{t1}, A_{t1})\}$, and the current question $Q_t$ at round t. The task is to rank a list of 100 candidate answers $A = \{A_1, A_2, ..., A_{100}\}$ and return the best answer $A_t$ for each question $Q_t$.

**Evaluation Metrics**

For the evaluation of response performance, a set of different metrics is used, which are suggested in [80]. Specifically, the model is required to return a sorting of 100 candidate answer options and gets evaluated on the following retrieved metrics:

1. **Existence of the human response in top k responses, i.e. Recall@k (R@k, k = 1, 5, 10)**. As stated above, the model ranks the 100 candidate answers from the set of possible answers $A = \{A_i, i = 0, \cdots, 99\}$. Recall@k metric depicts if the ground truth answer was between the first $k$ answers in the sorted list of possible answers. The lower of value of $k$, the better score the model assigned to the correct answer, the closest it was to understanding the question and returning the correct answer. Clearly, Recall@1 depicts the number of examples the model actually returned the correct answer.

2. **Mean rank of human response (Mean)**. As the name suggests it is the mean rank of the correct answer in the sorted list of possible answers. Since the list is sorted from the answer with the higher probability the answer with the lowest, the lower the value of this metric, the better.

3. **Mean Reciprocal Rank (MRR)**. It is defined as:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{5.1}$$

   where $rank_i$ is the rank of the human response and $Q$ the set of all questions. To achieve the perfect score, $MRR = 1$, the model has to rank consistently the correct response as the first answer.

4. **Normalized Discounted Cumulative Gain (NDCG)**, for a more comprehensive performance study. It uses dense annotations, i.e., the entire set of candidate answers is annotated as true or wrong. The metric penalizes low ranking correct answers, a method which addresses the issue when the set of answers contains more than one plausible result.

## 5.3 Related Work

As proposed by [72], a suitable model could follow the encoder-decoder framework. Specifically, the model consists of an encoder that maps the input (I, H, Qt) into a vector space, and a decoder that converts the embedded vector into an output.

Along with the VisDial dataset, Das et al. [72] proposed three baseline encoders. **Late Fusion (LF) Encoder**, which deals the dialog history as a concatenation of all the dialog rounds. Two different LSTMs[36] are used in order to encode the current question and the history. The individual representations of participating inputs (Image, History, current Question) are concatenated and linearly transformed to a desired size of joint representation. **Hierarchical Recurrent Encoder (HRE)**, which introduces a hierarchical encoding approach to capture the history. Lastly, **Memory Network (MN) Encoder**, maintains each previous question and answer as a 'fact' in its memory bank and learns to refer to the stored facts and image, in order to answer the question.

Apart from the aforementioned baselines, [81] utilizes recursive visual attention, inferring the co-reference through recursively inspecting the history dialog and improving the visual attention. [82] propose an EM-style inference algorithm to obtain the latent relations among history dialogs. MCA [83] focuses on an iterative question-conditioned context-aware graph, including both fine-grained visual and history semantics. DualVD [84] constructs a scene graph to represent the image, which emphasizes the essential role of vision for the referred visual content may change remarkably. DMRM [85] adopts multi-step reasoning based on dual attention to iteratively update related visual objects for a more relevant response. DAM [86] designs an adaptive decoder with memory to store the state of dialog history and visual information. Lastly, impressive results have been, also, achieved by pretrained models. Specifically, VisualBERT [87] and VDBERT [88] exploit large extra datasets to explore in visual dialog via pretraining language models.

**Visual Dialog with Graph Neural Networks**: Apart from the above approaches, there have been also several attempts to approach the visual dialog task using graph-based works. Specifically, [22] approaches the problem by focusing on the dialogue structure recovering, [23] and [24] on deep visual understanding and [25] on answer information revolving. The above works experiment on how the problem is modeled as a graph, e.g. only one graph for the whole dialog, or two graphs, one for each modality, what each graph node represents, e.g. a dialog round, a visual object, or on a lower level a token of each dialog round [25], as well as one the aggregation and propagation step the Graph Neural Network utilized. Finally, [26] suggests the use of ConceptNet to supply external knowledge and boost the model's performance.

## 5.4 Baseline model

As a baseline model we select the approach proposed in [12]. This approach suggests an interesting fusion of the two modalities and results to promising, announced by the authors, results. The model, *Knowledge-Bridge Graph Network (KBGN)* attempts to bridge the cross-modal semantic relations between vision and text modality, as well as to retrieve useful information via an adaptive information selection mode. Additionally, the reasoning clues can be clearly drawn from intra-modal entities and inter-modal bridges.

The KBGN framework consists of two graphs; The vision knowledge graph and the vision knowledge graph. The vision knowledge graph is a fully connected graph where the nodes $V = \{v_i | i\epsilon\{0, N-1\}\}$ are the visual entities, which are detected by a pre-trained Faster-RCNN [9] where N is the number of the detected in the image objects. The edges $E = \{e_{ij} | i, j\epsilon\{0, N-1\}\}$ are the visual relationships between nodes, extracted by a visual relationship encoder [89]. The text knowledge graph is also a fully connected graph, whose nodes $S = \{s_i | i\epsilon\{0, T-1\}\}$ represent each question-answer pair of the dialog history $H_t$, where t is the round number. For the text knowledge graph the initial node states are extracted by an bidirectional LSTM, which takes as input the concatenation of GloVe [70] and ELMo [71] word embeddings. The edges $R = \{r_{ij} | i, j\epsilon\{0, T-1\}\}$ represent the context dependence between the nodes and are computed by simply concatenating the two nodes, i.e. $r_{ij} = [s_i, s_j]$.

KBGN model consists of three main modules:

- **Knowledge Encoding**, which is applied independently to both modalities. By encoding the visual and textual information from the input using the current question as a guidance, this module aims to capture the text and the vision relations within each modality. To achieve that, it firstly applies a *Query-Guided Relation Selection*:

$$a_{ij} = softmax(W_e(W_1 Q_t \circ W_2 e_{ij})) \tag{5.2}$$

$$\hat{e}_{ij} = a_{ij} e_{ij} \tag{5.3}$$

Then updates the states of each node applying a *Query-Guided Graph Convolution*:

$$\beta_{ij} = softmax(W_v(Q_t \circ W_3[v_j, \hat{e}_{ij}])) \tag{5.4}$$

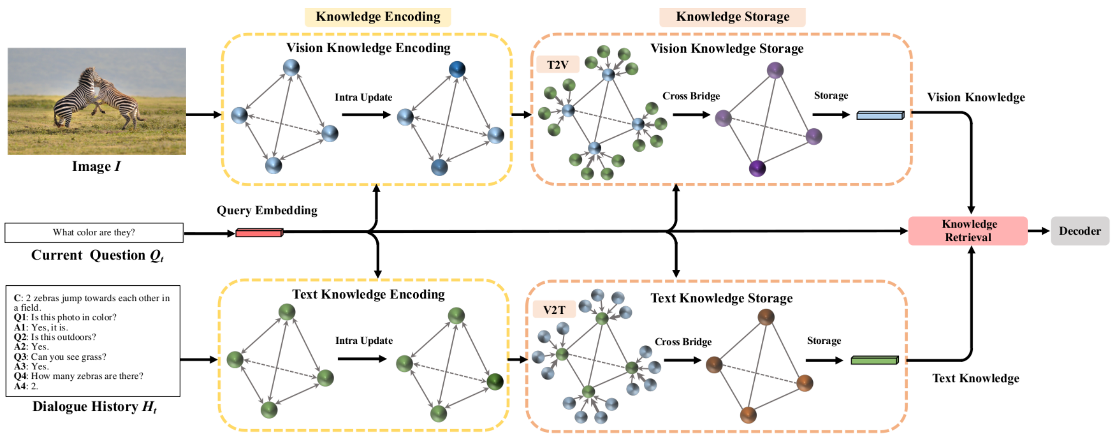


**Figure 5.3.** *Our main baseline: Knowledge-Bridge Graph Network [12]*

$$\hat{v}_i = \sum_{j=1}^{N} \beta_{ij} u_j \tag{5.5}$$

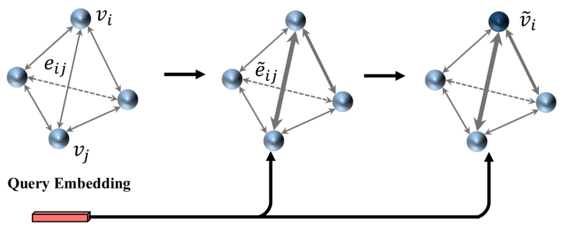


**Figure 5.4.** *Knowledge Encoding module in Knowledge-Bridge Graph Network [12]*

- **Knowledge Storage**, which aims to construct a bridging relation between image and text modality. Towards this goal, this module enriches the text modality with image and vice versa, the image modality with text information. Each node (center node) from each modality, is enriched by every node (cross node) of the other modality. For enriching the vision modality with the text information, the below procedure is followed.

  1. Construct Text to Vision GNN, where each intra-modal center node $v_i$ is connected with all the inter-modal cross nodes $\hat{S} = \hat{s}_i^t$ by t edges $B_{ij}^v$, t being the number of rounds. The edges capture underlying semantic dependence from each entity of text knowledge to vision knowledge, which is provided as $B_{ij}^v = [\hat{v}_i, \hat{s}_j]$
  2. Update the edges (bridges), under the guidance of the question:

$$\gamma_{ij} = softmax(W_b^v(W_4 Q_t \circ W_5 B_{ij}^v)) \tag{5.6}$$

$$\hat{B}_{ij}^v = \gamma_{ij} B_{ij}^v \tag{5.7}$$

  3. Update the vision nodes (center nodes) as:

$$\delta_{ij} = softmax(W_c^v(Q_t \circ W_6[s_j, \hat{B}_{ij}^v])) \tag{5.8}$$

$$\hat{v}_i^c = \sum_{j=1}^{t} \delta_{ij} \hat{s}_j \tag{5.9}$$

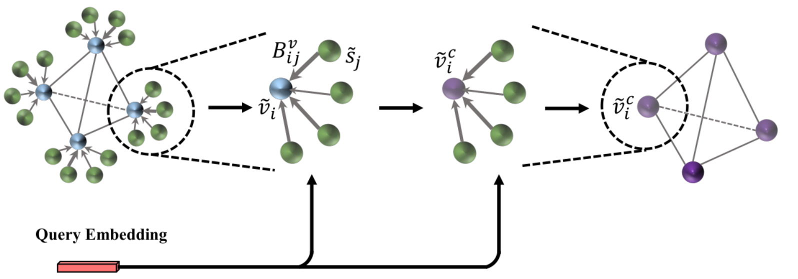


**Figure 5.5.** *Knowledge Storage's steps 2 and 3, where the edges between a vision node (blue) and the text nodes (green) are updated and then the each vision node is updated through a Query-Guided Cross Graph Convolution resulting in the updated visual nodes (purple). Source: [12]*

4. Apply a gate operation between the updated nodes of the knowledge encoding module, intra-dependence-aware local knowledge $\hat{v}_i$ and the updated nodes of the Text-to-Vision graph obtained in the previous step, inter-dependence-aware local knowledge $\hat{v}^c_i$:

$$gate^v_l = \sigma(W^c_l[\hat{v}_i, \hat{v}^c_i]) \tag{5.10}$$

$$\hat{v}^l_i = W_7(gate^v_l \circ [\hat{v}_i, \hat{v}^c_i]) \tag{5.11}$$

5. Obtain a global intra-modal knowledge information, by extracting a graph embedding of the updated graph produced in the knowledge encoding module:

$$\eta^v_i = softmax(W^v_e(Q_t \circ (W_8\hat{v}_i))) \tag{5.12}$$

$$\hat{I}_o = \sum_{i=1}^{N} \eta^v_i \hat{u}_i \tag{5.13}$$

6. Obtain the cross global intra-modal knowledge $\hat{I}_c$, , by extracting a graph embedding of the updated Text-to-Vision graph, with the guidance of the current question:

$$\mu^v_i = softmax(W^v_\alpha(Q_t \circ (W_9\hat{v}^l_i))) \tag{5.14}$$

$$\hat{I}_c = \sum_{i=1}^{N} \mu^v_i \hat{u}^l_i \tag{5.15}$$

7. Finally, obtain the global vision knowledge, using by a gate operation:

$$gate^v_g = \sigma(W^g_l[\hat{I}_o, \hat{I}_c]) \tag{5.16}$$

$$\hat{I} = W_{10}(gate^v_g \circ [\hat{I}_o, \hat{I}_c]) \tag{5.17}$$

The above procedure will be the same for the enrichment of the text modality using the visual information. Specifically, a Vision to Text GNN will be constructed, where each intra-modal center node $\hat{s}_i$ is connected with all the inter-modal cross nodes $\hat{V} = \hat{v}^N_i$ by N edges $B^s_{ij}$, defined as $B^s_{ij} = [\hat{s}_i, \hat{v}_j]$. The above steps are then applied using the appropriate center and cross nodes, resulting in the global text knowledge information $\hat{H}_t$, t being the current round number.

- **Knowledge Retrieval** module adaptively selects relative information from vision and text

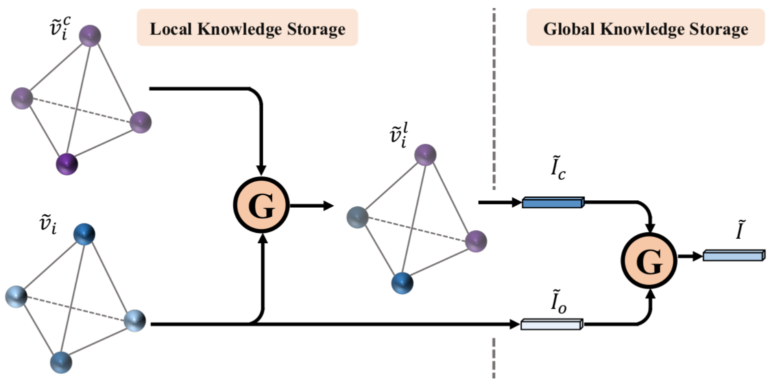


**Figure 5.6.** *Knowledge Storage's steps 4, **5**, **6** and **7**, where the "G" represents a gate operation. Source: [12]*

knowledge for the final answer:

$$gate_r = \sigma(W_r[Q_t, \hat{I}, \hat{H}_t]) \tag{5.18}$$

$$\hat{K} = W_{11}(gate_r \circ [Q_t, \hat{I}, \hat{H}_t]) \tag{5.19}$$

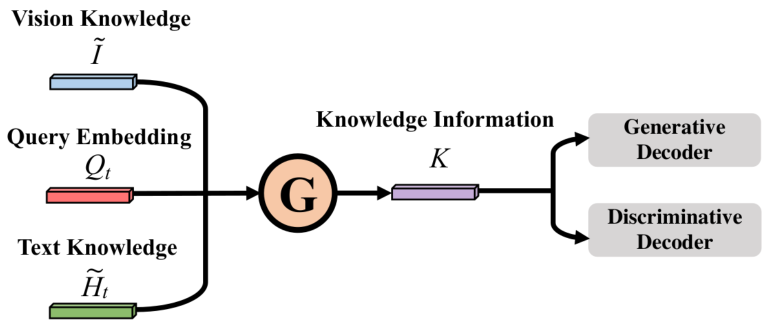


**Figure 5.7.** *Knowledge Retrieval module with multi-type decoders. "G" represents gate operation. Source: [12]*

The results of various models, as well as of the selected baseline, are shown in the below table:

| Results on test split | | | | | | |
|---|---|---|---|---|---|---|
| | MRR ↑ | R@1 ↑ | R@5 ↑ | R@10 ↑ | Mean ↓ | NDCG ↑ |
| LF[72] | 55.42 | 40.95 | 72.45 | 82.83 | 5.95 | 45.31 |
| HRE[72] | 54.16 | 39.93 | 70.47 | 81.50 | 6.41 | 45.46 |
| MN[72] | 55.49 | 40.98 | 72.30 | 83.30 | 5.92 | 47.50 |
| VGNN [22] | 61.37 | 47.33 | 77.98 | 87.83 | 4.57 | 52.82 |
| FGA [25] | 63.70 | 49.58 | 80.98 | 88.55 | 4.51 | 52.10 |
| DualVD [84] | 63.23 | 49.25 | 80.23 | 89.70 | 4.11 | 56.32 |
| CAG [23] | 63.49 | 49.85 | 80.63 | 90.15 | 4.11 | 56.64 |
| KBGN | 64.13 | 50.47 | 80.70 | 90.16 | 4.08 | 57.60 |

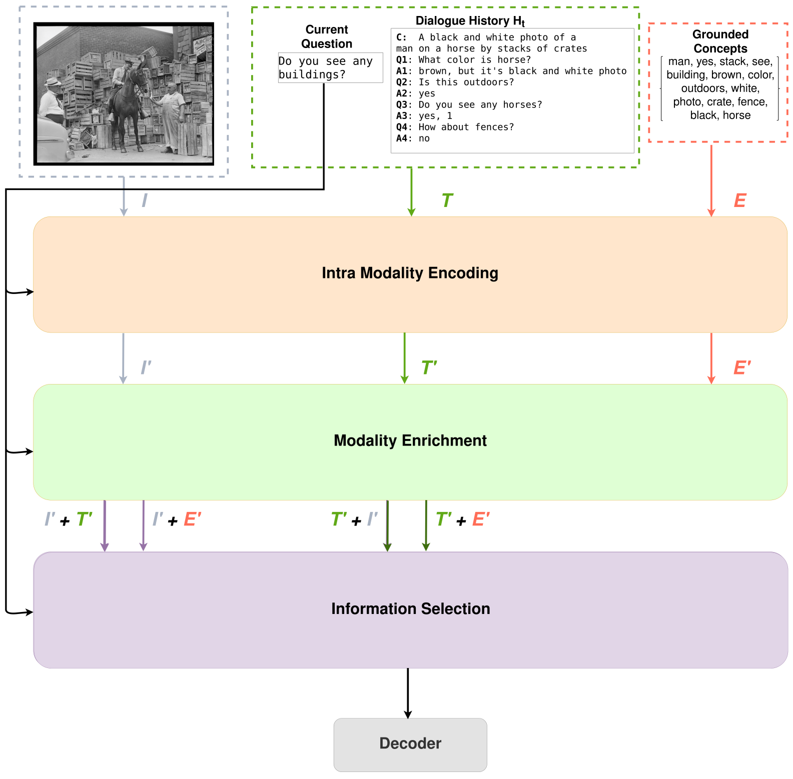
**Table 5.3.** *Results of baseline and various successful models on test-standard set of VisDial v1.0 on discriminative method.*

## 5.5 Proposed Method

Utilizing the fusion methods of the two modalities described in the above section, we attempt to exploit external knowledge gathered from the popular knowledge graph Concept-Net [60]. For this purpose a third graph, called External Knowledge Graph, was introduced in the model. Experiments were conducted on both how to better exploit the External Knowledge Graph and on the final fusion of the three modalities. In the below subsections the details of the implemented model are presented.

### 5.5.1 Core Architecture

The core idea of our approach is the introduction of a third modality, the *External Knowledge* modality and the fusion of it with the other two modalities, Vision and Text. The core procedure can be decomposed into three main modules, similarly to the baseline. Specifically, after encoding the information within each of the modalities, we enrich the Vision and the Text utilizing information derived from all the others. The result of it is two types of enriched information for the Vision and for the Text modality. Then, we combine the two enriched knowledges for each modality resulting in a complete knowledge for each modality. Finally, we combine the current Question with the Vision and the Text knowledge and feed the result to the decoder. An abstract scheme of our model is shown below, in Fig. 5.8:



Current Question
Do you see any buildings?
Dialogue History $H_t$
C: A black and white photo of a man on a horse by stacks of crates
Q1: What color is horse?
A1: brown, but it's black and white photo
Q2: Is this outdoors?
A2: yes
Q3: Do you see any horses?
A3: yes, 1
Q4: How about fences?
A4: no
Grounded Concepts
man, yes, stack, see, building, brown, color, outdoors, white, photo, crate, fence, black, horse
I
T
E
Intra Modality Encoding
I'
T'
E'
Modality Enrichment
I' + T'
I' + E'
T' + I'
T' + E'
Information Selection
Decoder

**Figure 5.8.** *Abstract visualisation of the proposed model*

Inspired by the baseline's architecture, we propose a model as shown in fig 5.9. Specifically, the model now has three different graphs, the visual, the text and the external knowledge graph. Each of these graphs represents a different modality and is encoded by the modules shown in the abstract figure.

The Vision and Text graph are constructed as described in the baseline section. The Vision

Graph is a fully connected graph:

$$G_V = (V, E) \tag{5.20}$$

$$V = \{v_i | i\epsilon\{0, N-1\}\} \tag{5.21}$$

$$E = \{e_{ij} | i, j\epsilon\{0, N-1\}\} \tag{5.22}$$

where $v_i \epsilon \mathbf{R^m}$, $e_{ij} \epsilon \mathbf{R^n}$, m is the image features dimensions, n is the dimensions of the visual relations extractor and N is the number of the detected in the image objects.

The Text Graph is also a fully connected graph as follows:

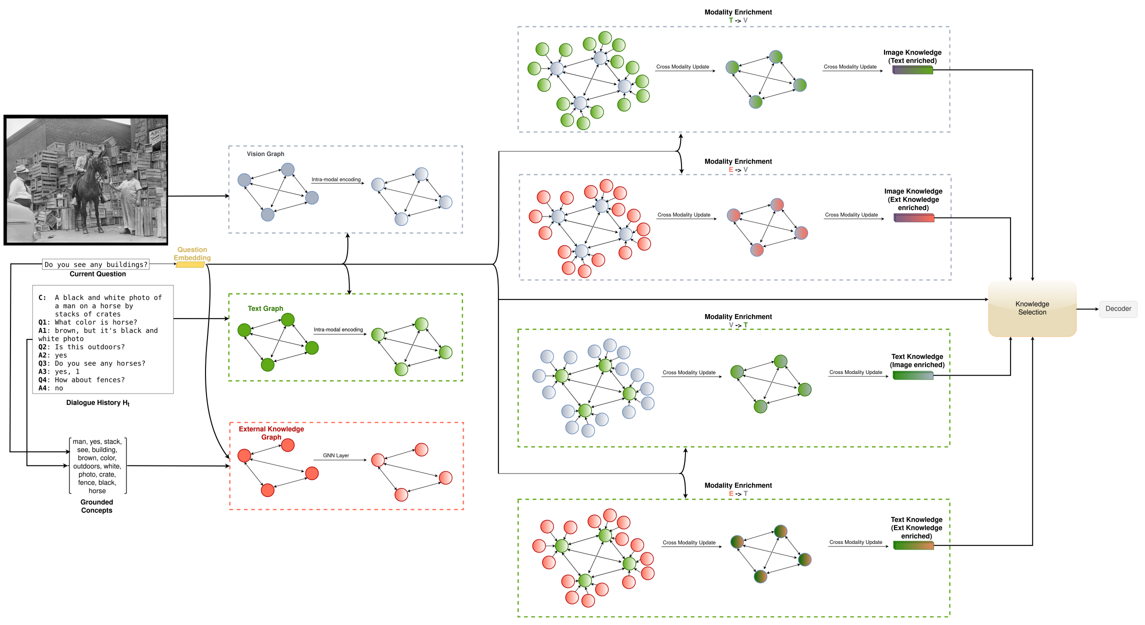
$$G_T = (S, R) \tag{5.23}$$

$$S = \{s_i | i\epsilon\{0, T-1\}\} \tag{5.24}$$

$$R = \{r_{ij} | i, j\epsilon\{0, T-1\}\} \tag{5.25}$$

where $s_i \epsilon \mathbf{R^k}$, $r_{ij} \epsilon \mathbf{R^2 k}$, k is the dimension of the LSTM hidden state and T is the number of rounds.

The External Knowledge graph has significant differences with the Vision and Text graphs, that lead to the need of a different encoding. Specifically, as a subset of the Concept-Net knowledge graph, it is a highly sparse graph in contrast to the vision and text graphs, which are fully connected. Moreover, each node in the External Knowledge represents a concept of the dialog and is initialized with the corresponding numberbatch embeddings. Finally, as described later in the preprocessing of the ConceptNet in section 7.2.2, there are multiple types of relationships present in the External Knowledge graph and taking all of them into account or not is a crucial decision, as shown by the experiments that followed.


Do you see any buildings?

C: A black and white photo of a man on a horse by stacks of crates
Q1: What color is horse?
A1: brown, but it's black and white photo
Q2: Is this outdoors?
A2: yes
Q3: Do you see any horses?
A3: yes, 1
Q4: How about fences?
A4: no

man, yes, stack, see, building, brown, color, outdoors, white, photo, crate, fence, black, horse

**Figure 5.9.** *The proposed model*

## Intra-modal encoding

For the vision and text graph encoding as in the baseline is applied independently for the two modalities. Using the question as a guidance for the encoding, we are able to capture the text and the vision relations within each modality. As described above, the encoding of the Vision Graph is as follows:

1. Relation update:
$$a_{ij} = softmax(W_e(W_1 Q_t \circ W_2 e_{ij})) \tag{5.26}$$
$$\hat{e}_{ij} = a_{ij} e_{ij} \tag{5.27}$$

2. Node update:
$$\beta_{ij} = softmax(W_v(Q_t \circ W_3[u_j, \hat{e}_{ij}])) \tag{5.28}$$
$$\hat{v}_i = \sum_{j=1}^{N} \beta_{ij} u_j \tag{5.29}$$

For the resulted Vision Graph node representation: $\hat{v}_i \epsilon \mathbf{R^m}$. The above equations are applied the same way for the Text Graph, obtaining the updated Text Graph node representations: $\hat{s}_i \epsilon \mathbf{R^k}$.

For the intra-modal encoding of the External Knowledge graph we experimented with various popular GNN layers. Specifically, several experiments were conducted, that demonstrated the effect of taking into account the relation types as well as applying self attention to the nodes. All these approaches are further discussed in Subsection 5.5.2.

The encoding of the External Knowledge Graph will result in the external knowledge information that together with the vision and text knowledge will be fed into the next modules.

### Modality enrichment

The modality enrichment module was derived from the Knowledge Storage module of the baseline model. Specifically, we introduce an abstract of the specific module, using center and cross nodes. The former belong to the modality that is being enriched, while the latter to the modality that is used to enrich the former.

Assuming we want to enrich modality M1, represented by graph $G_1 : (V_1, E_1)$ with modality M2, represented by graph $G_2 : (V_2, E_2)$, the following procedure is followed:

1. Construct M2-to-M1 GNN, where each intra-modal center node $v_{1_i}$ is connected with all the inter-modal cross nodes $v_2 \epsilon V_2$ by edges $B^1_{ij}$, provided as $B_{ij} = [v_{1_i}, v_{2_j}]$, where $i \epsilon 0, |V_1| - 1$ and $j \epsilon 0, |V_2| - 1$

2. Update the edges (bridges), under the guidance of the current question $Q_t$:
$$\gamma_{ij} = softmax(W_3(W_1 Q_t \circ W_2 B_{ij})) \tag{5.30}$$
$$\hat{B}_{ij} = \gamma_{ij} B_{ij} \tag{5.31}$$

3. Update the center nodes as:
$$\delta_{ij} = softmax(W_5(Q_t \circ W_6[v_{2_j}, \hat{B}_{ij}])) \tag{5.32}$$
$$\hat{v}^1_i = \sum_{j=1}^{|V_2|} \delta_{ij} v_{2_j} \tag{5.33}$$

As shown in Fig. 5.10, the nodes of $G_2$ graph (brown nodes) are used to enrich the information of $G_1$'s nodes (blue).

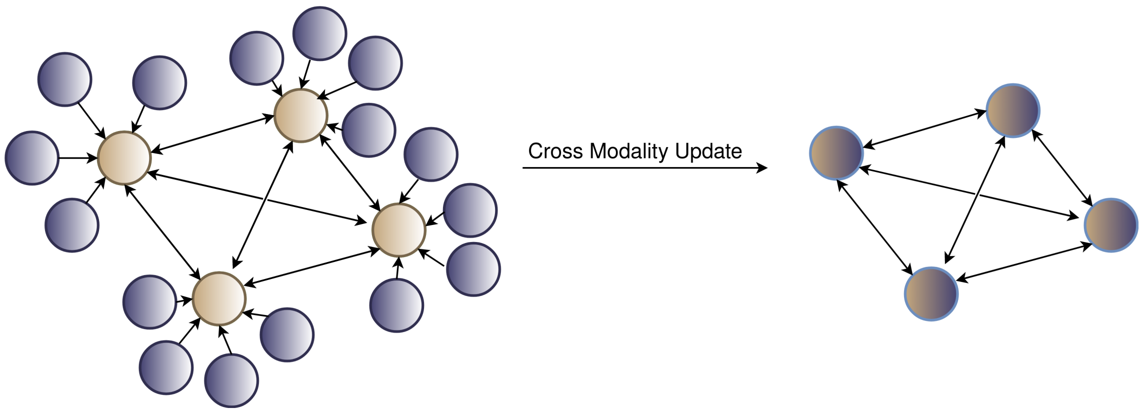


**Figure 5.10.** *Update of the center nodes $V_1$ (brown) by summing all the cross nodes $V_2$ (blue).*

4. Apply a gate operation between the input center nodes and the updated center nodes obtained in the previous step, inter-dependence-aware local knowledge $\hat{v}_i^1$:

$$gate_l = \sigma(W_7[v_{1_i}, \hat{v}_{1_i}]) \tag{5.34}$$

$$\hat{v}_{1_i}^l = W_8(gate_l \circ [v_{1_i}, \hat{v}_{1_i}]) \tag{5.35}$$

5. Obtain a global intra-modal knowledge information, by extracting a graph embedding of the updated graph produced in the knowledge encoding module:

$$\eta_i = softmax(W_9(Q_t \circ (W_{10}\hat{v}_{1_i}))) \tag{5.36}$$

$$\hat{V}_{1_o} = \sum_{i=1}^{|V_1|} \eta_i \hat{u}_{1_i} \tag{5.37}$$

6. Obtain the cross global intra-modal knowledge $\hat{I}_c$, , by extracting a graph embedding of the updated M2-to-M1 graph, with the guidance of the current question:

$$\mu_i = softmax(W_{11}(Q_t \circ (W_{12}v_{1_i}^l))) \tag{5.38}$$

$$\hat{V}_{1_c} = \sum_{i=1}^{|V_1|} \mu_i \hat{v}_{1_i}^l \tag{5.39}$$

7. Finally, obtain the global knowledge, using by a gate operation:

$$gate_g^{v_1} = \sigma(W_{13}[\hat{V}_{1_o}, \hat{V}_{1_c}]) \tag{5.40}$$

$$\hat{V}_1 = W_{14}(gate_g^{v_1} \circ [\hat{V}_{1_o}, \hat{V}_{1_c}]) \tag{5.41}$$

By the end of the above procedure we will have obtained the knowledge relevant to the M1 modality enriched by the M2 modality, which is represented by $\hat{V}^1$.

Using the above procedure, we employ four Modality enrichment modules:

1. Modality_enrichment(Vision, Text), to enrich the vision modality with information from the text modality. Results in $\hat{I}^T \epsilon \mathbf{R^k}$

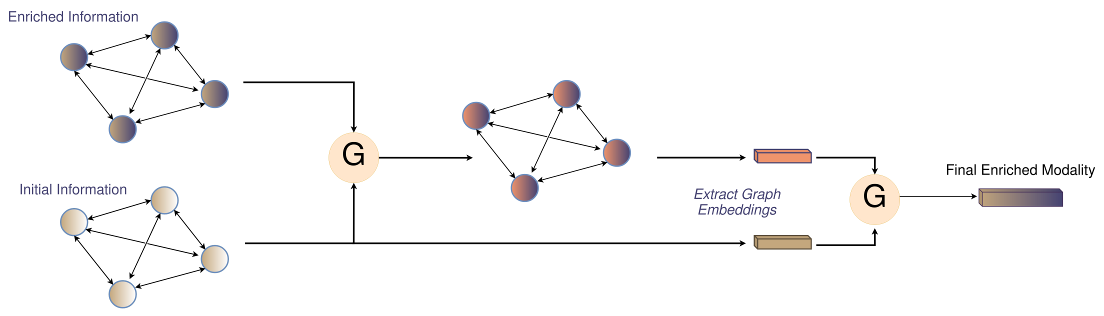


**Figure 5.11.** *Apply gate operations firstly one node level and then on graph level to acquire the final enriched information.*

2. Modality_enrichment(Vision, External Knowledge), to enrich the vision modality with information from the External Knowledge modality. Results in $\hat{I}^E \epsilon \mathbf{R^k}$

3. Modality_enrichment(Text, Vision), to enrich the text modality with information from the vision modality. Results in $\hat{T}^I \epsilon \mathbf{R^k}$

4. Modality_enrichment(Text, External Knowledge), to enrich the text modality with information from the External Knowledge modality. Results in $\hat{T}^E \epsilon \mathbf{R^k}$

The above result into four different sources of knowledge, that will be fed into the Knowledge Retrieval module.

### Knowledge Selection

The Knowledge Selection module is the original module from the baseline, modified in order to adaptively select relative information from the four sources of knowledge; vision enriched by text $\hat{I}^T$ , vision enriched by external knowledge $\hat{I}^E$,text enriched by vision $\hat{T}^I$, text enriched by external knowledge $\hat{T}^E$. Specifically, we extract a global vision knowledge, enriched by the text and the external knowledge modality and a global text knowledge, enriched by the vision and the external knowledge modality. Finally, we select the relevant information, with a method similar to the one in the baseline model:

$$gate_i = \sigma(W_r[\hat{I}^T, \hat{I}^E]) \tag{5.42}$$

$$\hat{K}_I = W_{15}(gate_r \circ [\hat{I}^T, \hat{I}^E]) \tag{5.43}$$

$$gate_T = \sigma(W_r[\hat{T}^I, \hat{T}^E]) \tag{5.44}$$

$$\hat{K}_T = W_{16}(gate_r \circ [\hat{T}^I, \hat{T}^E]) \tag{5.45}$$

$$gate_r = \sigma(W_r[Q_t, \hat{K}_I, \hat{K}_T]) \tag{5.46}$$

$$\hat{K} = W_{17}(gate_r \circ [Q_t, \hat{K}_I, \hat{K}_T]) \tag{5.47}$$

For the above results: $\hat{K}_I \epsilon \mathbf{R^k}$, $\hat{K}_T \epsilon \mathbf{R^k}$, $\hat{K} \epsilon \mathbf{R^k}$. The Knowledge Selection is shown in Fig. 5.12.

## 5.5.2 GNNs for external Graph

The below GNN variants will be used to encode the External Knowledge Graph:

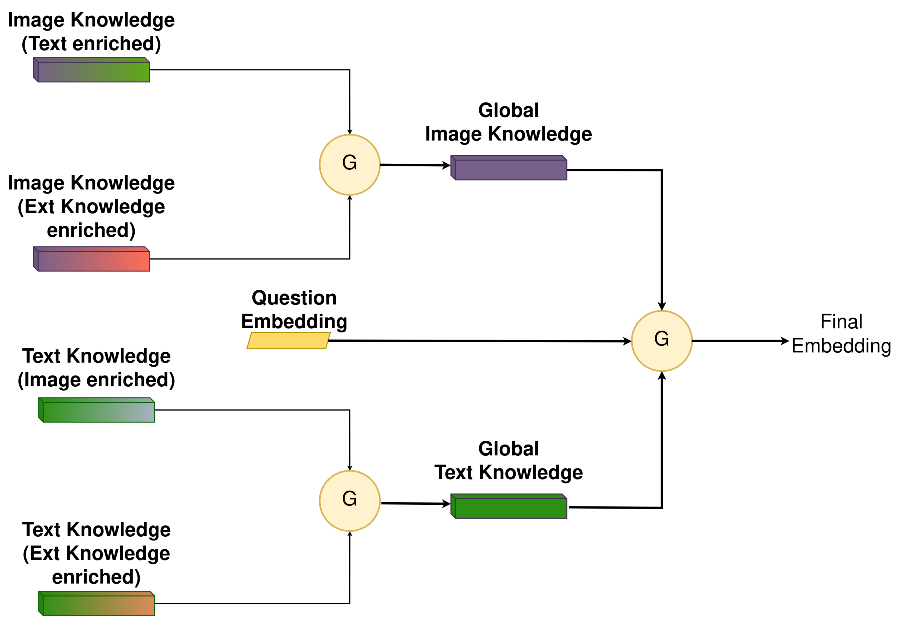


Image Knowledge (Text enriched)
Image Knowledge (Ext Knowledge enriched)
G
Global Image Knowledge
Question Embedding
G
Final Embedding
Text Knowledge (Image enriched)
Text Knowledge (Ext Knowledge enriched)
G
Global Text Knowledge

**Figure 5.12.** *Apply gate operations on the two types of knowledge for each modality and then one final gate operation between the two modalities to extract the final embedding.*

### GCN [14]

As Graph Convolution Networks do not take into account multiple types of relations, this layer considers only one type. The relationship *related_to* was selected for this purpose.

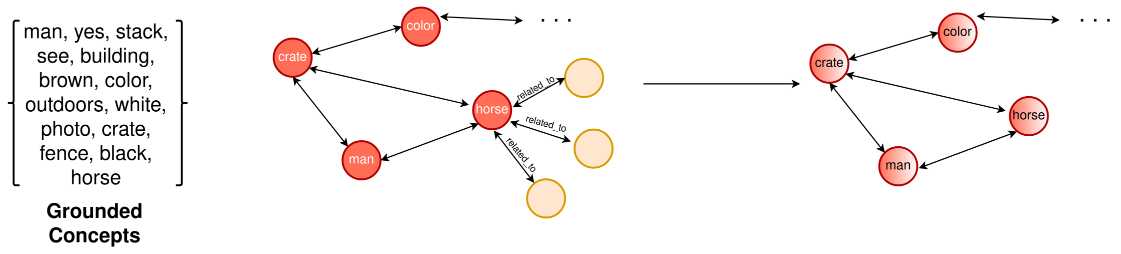

man, yes, stack, see, building, brown, color, outdoors, white, photo, crate, fence, black, horse
Grounded Concepts
color
crate
horse
man
related_to
related_to
related_to
. . .
color
crate
horse
man
. . .

**Figure 5.13.** *GCN for encoding the External Knowledge Graph.*

In order to guide the GCN using the current question, two different methods where used:

- Concatenating the question:

$$h_v^k = \sigma(W_k \sum_{u(v)} \frac{[h_v^{k-1}, Q_t]}{deg(v)}) \tag{5.48}$$

- Modifying the neighbouring nodes before the aggregation, by computing coefficients responsible for emphasizing only to relevant with the question information:

$$h_{vq}^{k-1} = softmax(W_2(Q_t \circ W_1 h_v^{k-1})) * h_v^{k-1} \tag{5.49}$$

$$h_v^k = \sigma(W_k \sum_{u(v)} \frac{h_{vq}^{k-1}}{deg(v)}) \tag{5.50}$$

### R-GCN[27]

The typical layer for encoding knowledge graphs is R-GCN, as this type of GNN respects the types of the relations:

$$h_i^{(l+1)} = \sigma(\sum_{r \epsilon R} \sum_{j \epsilon N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)}) \tag{5.51}$$

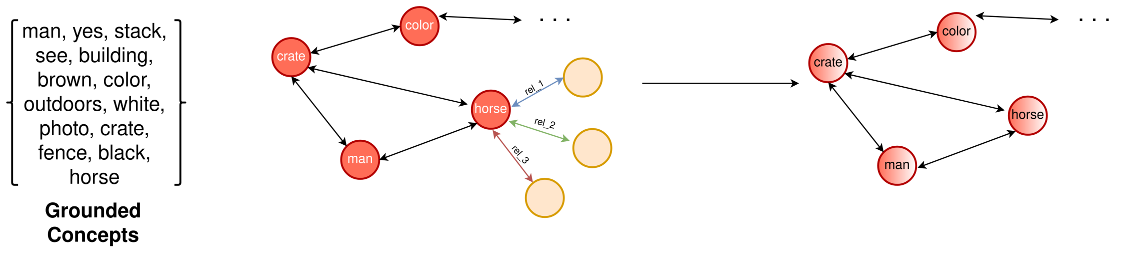

man, yes, stack, see, building, brown, color, outdoors, white, photo, crate, fence, black, horse
Grounded Concepts
color
crate
horse
man
rel_1
rel_2
rel_3
...
color
crate
horse
man
...

**Figure 5.14.** *R-GCN for encoding the External Knowledge Graph.*

To address the issue very varying frequency for the different relations, explained in Section 7.2.2, we employ *basis-decomposition* as proposed by [27], defining each weight $W_r$ as:

$$W_r^{(l)} = \sum_{b=1}^{B} a_{rb}^{(l)} V_b^{(l)} \tag{5.52}$$

This suggests that the weights are computed as a linear combination of basis transformations $V_b^{(l)} \epsilon \mathbf{R}^{d^{(l+1)} x d^{(l)}}$, with coefficients $a_{rb}^{(l)}$ such that only the coefficients depend on the relation $r$. Basis decomposition is in practice a weight sharing between different relation types, which significantly reduces the parameters needed to learned multi-relational data, such as a knowledge graph. This weight sharing also helps tackling overfitting on rare relations, since parameters updates are shared between all relations.

### GAT[4]

Graph Attention Network applies self-attention between all the nodes. Like GCN, it is able to consider only one kind of relation. As also shown in a previous section, the updated node embeddings are computed as:

$$a_{ij} = \frac{exp(LeakyReLU(a^T[Wh_i \parallel Wh_j]))}{\sum_{k \epsilon N_i} exp(LeakyReLU(a^T[Wh_i \parallel Wh_k]))} \tag{5.53}$$

$$h_i' = \sigma(\sum_{j \epsilon N_i} a_{ij} W h_j) \tag{5.54}$$

$a_{ij}$ is the attention coefficient of node j to node i, while $h_i$ is the output features of each node.

As proposed in the paper, we employ multi-head attention. Specifically, K (K=8) independent attention mechanisms execute the above equation and then their features are concatenated, resulting in the following output feature representation:

$$h_i' = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \epsilon N_i} a_{ij}^k W^k h_j) \tag{5.55}$$

Finally, in order to use the current question $Q_t$ to guide the procedure, we concatenating the question embedding with the embeddings of the nodes. As a result the above equations became:

$$a_{ij} = \frac{exp(LeakyReLU(a^T[W[h_i \parallel Q_t] \parallel W[h_j \parallel Q_t]]))}{\sum_{k \epsilon N_i} exp(LeakyReLU(a^T[W[h_i \parallel Q_t] \parallel W[h_k \parallel Q_t]]))} \tag{5.56}$$

$$h'_i = \sigma(\sum_{j \epsilon N_i} a_{ij} W[h_j \parallel Q_t]) \tag{5.57}$$

$$h'_i = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \epsilon N_i} a_{ij}^k W^k [h_j \parallel Q_t]) \tag{5.58}$$

**Custom message passing method**

We also experimented with a method to update the node embeddings of the External Knowledge Graph inspired from the Message Passing Framework [28]. Specifically, considering again only one type of relations, the *related_to* type, we represent the edges of the graph as the concatenation of the neighbouring nodes:

$$e_{ij} = [v_i, v_j] \tag{5.59}$$

where "[·, ·]" denotes concatenation.
We then update the edges using the current question $Q_t$ to construct the coefficients $a$:

$$a_{ij} = softmax(W_e(W_1 Q_t \circ W_2 e_{ij})) \tag{5.60}$$

$$\hat{e}_{ij} = a_{ij} e_{ij} \tag{5.61}$$

Finally, we apply a weighted sum on each node's neighbourhood similar to a GCN layer with coefficients defined as:

$$\beta_{ij} = softmax(W_v(Q_t \circ W_3[u_j, \hat{e}_{ij}])) \tag{5.62}$$

$$\hat{v}_i = \sum_{j=1}^{N} \beta_{ij} u_j \tag{5.63}$$

## 5.6 Experiments

Implementation details for the experiments described in the following subsections, as well as details on the preprocessing of the ConceptNet [60] Knowledge Graph, used as the source of the External Knowledge, are presented in the Appendix 7.

### 5.6.1 Discriminative Decoder

Discriminative decoder ranks all the answers in the answer candidates A.The decoder computes dot product similarity between the input encoding and an LSTM encoding of each of the answer options. These dot products are fed into a softmax to compute the posterior probability over the options. During training, we maximize the log- likelihood of the correct option. During evaluation, options are simply ranked based on their posterior probabilities.
Results on VisDial v1.0 dataset:

In the above tables we denote by *KBGN-Ext-GCN-CON-Q* and *KBGN-Ext-GCN-S-Q* the models described in 5.5.2. The former concatenates the question with the node embeddings, while

| Results on test split | | | | | | |
|---|---|---|---|---|---|---|
| | MRR ↑ | R@1 ↑ | R@5 ↑ | R@10 ↑ | Mean ↓ | NDCG ↑ |
| KBGN | 64.13 | 50.47 | 80.70 | 90.16 | 4.08 | 57.60 |
| KBGN-Impl | 62.82 | 48.95 | 80.33 | 89.23 | 4.28 | 56.62 |
| KBGN-Numb | 62.59 | 48.42 | 80.23 | 89.16 | 4.28 | 56.19 |
| KBGN-Ext-GCN-CON-Q | 63.25 | 49.43 | 80.05 | 89.63 | 4.21 | 55.79 |
| KBGN-Ext-GCN-S-Q | 62.78 | 48.95 | 79.48 | 89.0 | 4.32 | 55.57 |
| KBGN-Ext-GAT | 55.69 | 42.7 | 70.18 | 79.73 | 7.87 | 51.87 |
| KBGN-Ext-GAT-Q | 56.21 | 43.04 | 70.85 | 80.02 | 7.79 | 51.98 |
| KBGN-Ext-RGCN | 62.873 | 49.2 | 79.35 | 89.15 | 4.33 | 55.78 |
| KBGN-Ext-MessagePassing | 62.71 | 48.93 | 80.32 | 89.21 | 4.11 | 56.58 |

**Table 5.4.** *Result comparison on test-standard set of VisDial v1.0 on discriminative method.*

| Results on validation split | | | | | | |
|---|---|---|---|---|---|---|
| | MRR ↑ | R@1 ↑ | R@5 ↑ | R@10 ↑ | Mean ↓ | NDCG ↑ |
| KBGN | 64.86 | 51.37 | 81.71 | 90.54 | 4.00 | 59.08 |
| KBGN-Impl* | 63.84 | 50.04 | 80.87 | 90.02 | 4.104 | 57.056 |
| KBGN-Numb | 63.61 | 49.69 | 80.68 | 89.93 | 4.11 | 56.64 |
| KBGN-Ext-GCN-CON-Q | 64.04 | 50.38 | 81.16 | 89.92 | 4.15 | 56.58 |
| KBGN-Ext-GCN-S-Q | 63.60 | 49.73 | 80.66 | 89.97 | 41.15 | 55.74 |
| KBGN-Ext-GAT | 69.41 | 56.07 | 86.3 | 93.84 | 3.16 | 53.42 |
| KBGN-Ext-GAT-Q | 70.68 | 57.31 | 87.23 | 94.29 | 3.10 | 54.57 |
| KBGN-Ext-RGCN | 63.7 | 49.93 | 80.59 | 89.95 | 4.12 | 55.63 |
| KBGN-Ext-MessagePassing | 63.91 | 50.09 | 80.90 | 89.95 | 4.1 | 56.52 |

**Table 5.5.** *Result comparison on validation set of VisDial v1.0 on discriminative method.*

the latter uses the question to generate coefficients for the nodes of the graph. The models *KBGN-Ext-GAT* and *KBGN-Ext-GAT-Q* correspond to the models described in 5.5.2 without and with the question respectively. By *KBGN-Ext-RGCN* we denote the model described in 5.5.2 and by *KBGN-Ext-MessagePassing* the one in 5.5.2. Finally, *KBGN-Numb* refers to a model exactly as the baseline, but using the Numberbatch embeddings instead of the concatenation of GloVe and ELMo embeddings.

### 5.6.2 Generative Decoder

Generative decoder outputs probability distribution over the vocabulary at each decoding step. The encoded vector is set as the initial state of the Long Short-Term Memory (LSTM) RNN language model. During training, we maximize the log-likelihood of the ground truth answer sequence given its corresponding encoded representation (trained end-to-end). To evaluate, we use the model's log-likelihood scores and rank candidate answers. Since this decoder does not score options during training, models utilizing it do not take advantage of the biases in option creation and tend to underperform models that do. On the other hand, generative decoders are more practical in that they can actually be deployed in realistic applications, since in those cases there will not be a list of candidate answers.

### 5.6.3 Results Discussion

As a first general comment on the results, we can see that models that perform well with the one decoder, perform also well using the other one. The poorer results using the generative decoder, compared with the discriminative one, are expected since the former does not score options during

| Results on validation split | | | | | | |
|---|---|---|---|---|---|---|
| | MRR ↑ | R@1 ↑ | R@5 ↑ | R@10 ↑ | Mean ↓ | NDCG ↑ |
| KBGN | 50.05 | 40.40 | 60.11 | 66.82 | 17.54 | 60.42 |
| KBGN-Impl | 48.09 | 39.04 | 56.97 | 62.58 | 20.82 | 56.94 |
| KBGN-Numb | 47.88 | 38.74 | 56.62 | 62.03 | 20.95 | 56.42 |
| KBGN-Ext-GCN-CON-Q | 48.23 | 39.05 | 57.39 | 63.35 | 20.72 | 57.31 |
| KBGN-Ext-GCN-S-Q | 48.01 | 38.98 | 56.91 | 62.53 | 20.96 | 56.84 |
| KBGN-Ext-GAT | 51.01 | 41.73 | 60.75 | 66.58 | 19.18 | 55.46 |
| KBGN-Ext-GAT-Q | 51.36 | 42.03 | 61.38 | 66.97 | 19.06 | 55.58 |
| KBGN-Ext-RGCN | 48.15 | 39.03 | 57.01 | 62.89 | 20.78 | 56.27 |
| KBGN-Ext-MessagePassing | 48.12 | 38.93 | 56.95 | 63.19 | 21.78 | 57.25 |

**Table 5.6.** *Result comparison on val-standard set of VisDial v1.0 on generative method.*

training and as a result does not exploit the biases in option creation and typically underperform models that do, such as the latter. Although both decoders where proposed by the creators of the task of Visual Dialog [11], the majority of research works only evaluates on the discriminative one.

Based on the above tables 5.4 and 5.5 we can see that our results are comparable to the implemented baseline, but only a few manage to surpass it. Specifically, the model KBGN-Ext-GCN-CON-Q, while the simplest one, seems to achieve the best results and surpasses the implemented baseline by 0.43% on the test set. KBGN-Ext-RGCN also performs slightly better than the implemented baseline. One possible reason this model does not achieve better results is the very different frequencies of the types of relations. As shown in tables 7.3, 7.4 and 7.5 the relation *related_ to* is by far dominant in the External Knowledge Graph. During our experiments we excluded the more rare relations and used the relations of the second column in Table 7.2, but the imbalance is still very obvious. The KBGN-Ext-MessagePassing model, although it follows an approach similar to the encoding of the other two graphs of the Image and Text modality, does not achieve good results. This is possibly due to the fact that these graphs have very different structure.

# Chapter 6

# Conclusions

## 6.1 Discussion

In this work we investigate different methods, in order to deal with the task of Visual Dialog. Visual Dialog, along with other similar tasks that combine Natural Language Processing and Computer Vision, has benefited greatly from progress in these fields. More and more sophisticated methods contribute towards improved performances. While AI has still a lot of progress to make to create models comparable to humans in these kind of tasks, state-of-the-art results are being constantly surpassed by a lot researchers setting the bar even higher. Our approach utilizes Graph Neural Networks. This type of networks has been proved until now very promising in a vast variety of fields. Hence, it is gaining a lot of attention by the research community, as it is considered to have a lot to offer in the future. We, also, employed an External Knowledge Graph, the ConceptNet[60]. Towards introducing more information to the model and boost its performance, we treated the external information as a third modality in our model. We utilize three different graphs one for each modality and employ three stages of encoding and fusing them, while we mainly experimented with encoding of the External Knowledge Graph.

We evaluate our models using two different decoders, the discriminative and the generative decoder. By comparing the two decoders we conclude that models that perform well using the the one decoder, perform also well employing the other one. As explained in subsection 5.6.2, the generative decoder does not match the performance of the discriminative. This result is expected, since the former does not score options during training and as a result does not exploit the biases in option creation. Meanwhile, the consistent results of our models, using both decoders, highlights the impact of the different encoders.

Based on the results of our experiments we are able to draw some conclusions about the impact of the External Knowledge, as well as the methods of encoding it. Specifically, we manage to surpass the implemented by us baseline's results using the methods *KBGN-Ext-GCN-CON-Q* and *KBGN-Ext-RGCN*, as shown in tables 5.4, 5.5 and 5.6. The former approach took into account only the relation *related_ to*, while the latter multiple types of relations. This second method surpassed the results only with a slight difference, but we believe that a better tuning of the various hyper-parameters of this model could also lead to even better results. The methods applying self-attention on the External Graph's nodes, namely *KBGN-Ext-GAT* and *KBGN-Ext-GAT-Q*, achieve better results on the validation set, but very poor performance on the test split. This behaviour may indicate false tuning of some hyperparameters. The custom method *KBGN-Ext-MessagePassing*, which is a more generic model with many more parameters could not achieve great results, while it led to increased number of parameters. Finally, *KBGN-Numb*, which is just the baseline using only Numberbatch embeddings achieved the worst results of all the experiments. The poor performance of the last experiment demonstrates that these embeddings alone, which are constructed from the ConceptNet, were inadequate to lead to better results and an External Knowledge Graph was necessary in order to introduce and exploit external information.

All in all, we conclude that the introduction of the External Knowledge was beneficial for the overall performance of the model. The best methods to encode the External Knowledge Graph were *KBGN-Ext-GCN-CON-Q* and *KBGN-Ext-RGCN*. These conclusions were highly expected. The former method is the one used for encoding the a graphs of external knowledge in tasks similar to ours, such as in [29]. The latter is a GNN layer designed for handling mutli-relational Graphs, such as Knowledge Graphs and as a result was expected to boost the model's preformance. As final note on the External Knowledge Graph, we have to emphasize the importance of a necessary preprocessing to the ConceptNet knowledge graph. The design choices of the algorithm as well as the desired functionality were crucial for creating a time efficient preprocessing and for fetching relevant to the dialog information respectively.

## 6.2 Future Work

By the end of this thesis, we desire to suggest some new paths towards more complex and interesting methods of modeling the data using graphs and fusing the modalities of our task. Comparing the various approaches against the Visual Dialog task, that employed Graph Neural Networks, we can see that these types of models enable us to handle the problem in many different ways. Starting with how we model the components of the task, almost each approach follows a different path. One could model the whole dataset into one big graph or into two, one for each modality. The nodes of the graph may represent tokens from the dialogue's sentences or even whole rounds of the dialog. Moreover, given a method of modeling the dataset, there exist a few different ways of fusing the two modalities. From a simple concatenation of the final representations, the graph embeddings of the two modalities (late fusion), which has been widely used as a fusion method, to more complex fusion techniques combining the graphs representing the modalities.

We encourage the following points be explored in future work:

- **Use embeddings to represent edges of the Knowledge Graph**: Instead of grouping all the r-neighbours for each different relation type $r$, as in R-GCN, one could experiment representing each relation type utilizing TransE [90] embeddings. By doing so, the model could have a better understanding of what are the relations connecting a node with different sets of other nodes.

- **Use LSTM for encoding the Knowledge Graph**: A very simple and straightforward way of encoding a Knowledge Graph is by using a Recurrent Neural Network, such as an LSTM, to encode the sequence: *"subject relation object"*, generated from the triplet *<subject, relation, object>* of the Knowledge Graph.

- **Design more sparse graphs**: As described in Section 5.5.1, the graphs that represent the Image and Text modality are fully connected. Utilizing the question to only connect relevant tokens from the dialog and image objects in the two graphs would increase the sparsity of those, result to a more meaningful representation of the modalities and decrease number of parameters. In the current work we used coefficients for each node, generated using a softmax function. In this way we filtered the irrelevant nodes, instead of not taking them at all into account (assigning them a coefficient equal to zero).

- **Multi-Step Reasoning** This method of computing attention weights more than once to focus even better on relevant information, was used for the task of Visual Dialog by [91] and achieved state-of-the-art results. Using a similar method together with Graph Neural Networks could be an insightful future work.

- **Temporal Graph Neural Networks ([56], [57])** These types of GNNs are able to handle graphs that change over time. Although they are currently being used mostly in traffic systems and social networks, they could be adapted to the Visual Dialog task, since the dialog is also a concept that alters over time. One approach could model the whole dialog using a big dynamic graph, that changes as the rounds of the dialog continue.

# Bibliography

[1] Esperanza García-Gonzalo, Zulima Fernández-Muñiz, Paulino García Nieto, Antonio Bernardo Sánchez και Marta Menéndez Fernández. *Hard-Rock Stability Analysis for Span Design in Entry-Type Excavations with Learning Classifiers. Materials*, 9(7):531, 2016.

[2] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner και G. Monfardini. *The Graph Neural Network Model. IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[3] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li και Razvan Pascanu. *Relational inductive biases, deep learning, and graph networks*, 2018. arXiv:1806.01261 [cs, stat].

[4] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò και Yoshua Bengio. *Graph Attention Networks. arXiv:1710.10903 [cs, stat]*, 2018. arXiv: 1710.10903.

[5] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov και Max Welling. *Modeling Relational Data with Graph Convolutional Networks. arXiv:1703.06103 [cs, stat]*, 2017. arXiv: 1703.06103.

[6] Xiaohan Zou. *A Survey on Application of Knowledge Graph. Journal of Physics: Conference Series*, 1487(1):012016, 2020.

[7] Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan και Xiang Ren. *Scalable Multi-Hop Relational Reasoning for Knowledge-Aware Question Answering*, 2020. arXiv:2005.00646 [cs].

[8] Tomas Mikolov, Kai Chen, Greg Corrado και Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*, 2013. arXiv:1301.3781 [cs].

[9] Shaoqing Ren, Kaiming He, Ross Girshick και Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Advances in Neural Information Processing Systems*, τόμος 28. Curran Associates, Inc., 2015.

[10] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár και C. Lawrence Zitnick. *Microsoft COCO: Common Objects in Context. Computer Vision – ECCV 2014*David Fleet, Tomas Pajdla, Bernt Schiele και Tinne Tuytelaars, επιμελητές, Lecture Notes in Computer Science, σελίδες 740–755, Cham, 2014. Springer International Publishing.

[11] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh και Dhruv Batra. *Visual Dialog. arXiv:1611.08669 [cs]*, 2017. arXiv: 1611.08669.

[12] Xiaoze Jiang, Siyi Du, Zengchang Qin, Yajing Sun και Jing Yu. *KBGN: Knowledge-Bridge Graph Network for Adaptive Vision-Text Reasoning in Visual Dialogue. Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, σελίδες 1265–1273, New York, NY, USA, 2020. Association for Computing Machinery.

[13] William L. Hamilton, Rex Ying και Jure Leskovec. *Inductive Representation Learning on Large Graphs. arXiv:1706.02216 [cs, stat]*, 2018. arXiv: 1706.02216.

[14] Thomas N. Kipf και Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907 [cs, stat]*, 2017. arXiv: 1609.02907.

[15] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell και Peter Battaglia. *Graph networks as learnable physics engines for inference and control. arXiv:1806.01242 [cs, stat]*, 2018. arXiv: 1806.01242.

[16] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende και Koray Kavukcuoglu. *Interaction Networks for Learning about Objects, Relations and Physics. arXiv:1612.00222 [cs]*, 2016. arXiv: 1612.00222.

[17] Alex Fout, Jonathon Byrd, Basir Shariat και Asa Ben-Hur. *Protein Interface Prediction using Graph Convolutional Networks.* σελίδα 10.

[18] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo και Yuji Matsumoto. *Knowledge Transfer for Out-of-Knowledge-Base Entities: A Graph Neural Network Approach. Transactions of the Japanese Society for Artificial Intelligence*, 33(2):F–H72_1–10, 2018. arXiv: 1706.05674.

[19] Hanjun Dai, Elias B. Khalil, Yuyu Zhang, Bistra Dilkina και Le Song. *Learning Combinatorial Optimization Algorithms over Graphs. arXiv:1704.01665 [cs, stat]*, 2018. arXiv: 1704.01665.

[20] Satwik Kottur, José M. F. Moura, Devi Parikh, Dhruv Batra και Marcus Rohrbach. *Visual Coreference Resolution in Visual Dialog Using Neural Module Networks. Computer Vision – ECCV 2018*Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu και Yair Weiss, επιμελητές, τόμος 11219, σελίδες 160–178. Springer International Publishing, Cham, 2018. Series Title: Lecture Notes in Computer Science.

[21] Jiasen Lu, Anitha Kannan, Jianwei Yang, Devi Parikh και Dhruv Batra. *Best of Both Worlds: Transferring Knowledge from Discriminative Learning to a Generative Visual Dialog Model. Advances in Neural Information Processing Systems*, τόμος 30. Curran Associates, Inc., 2017.

[22] Zilong Zheng, Wenguan Wang, Siyuan Qi και Song Chun Zhu. *Reasoning Visual Dialogs with Structural and Partial Observations. arXiv:1904.05548 [cs]*, 2019. arXiv: 1904.05548.

[23] Dan Guo, Hui Wang, Hanwang Zhang, Zheng Jun Zha και Meng Wang. *Iterative Context-Aware Graph Inference for Visual Dialog. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, σελίδες 10052–10061, Seattle, WA, USA, 2020. IEEE.

[24] Xiaoze Jiang, Jing Yu, Zengchang Qin, Yingying Zhuang, Xingxing Zhang, Yue Hu και Qi Wu. *DualVD: An Adaptive Dual Encoding Model for Deep Visual Understanding in Visual Dialogue. Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11125–11132, 2020. Number: 07.

[25] Idan Schwartz, Seunghak Yu, Tamir Hazan και Alexander Schwing. *Factor Graph Attention. arXiv:1904.05880 [cs]*, 2020. arXiv: 1904.05880.

[26] Lei Zhao, Lianli Gao, Yuyu Guo, Jingkuan Song και Hengtao Shen. *SKANet: Structured Knowledge-Aware Network for Visual Dialog. 2021 IEEE International Conference on Multimedia and Expo (ICME)*, σελίδες 1–6, 2021. ISSN: 1945-788X.

[27] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov και Max Welling. *Modeling relational data with graph convolutional networks. European semantic web conference*, σελίδες 593–607. Springer, 2018.

[28] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals και George E. Dahl. *Neural Message Passing for Quantum Chemistry*, 2017. arXiv:1704.01212 [cs].

[29] Bill Yuchen Lin, Xinyue Chen, Jamin Chen και Xiang Ren. *KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning. arXiv:1909.02151 [cs]*, 2019. arXiv: 1909.02151.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser και Illia Polosukhin. *Attention is All you Need. Advances in Neural Information Processing Systems*, τόμος 30. Curran Associates, Inc., 2017.

[31] Y. Lecun, L. Bottou, Y. Bengio και P. Haffner. *Gradient-based learning applied to document recognition. Proceedings of the IEEE*, 86(11):2278–2324, 1998. Conference Name: Proceedings of the IEEE.

[32] David E. Rumelhart, Geoffrey E. Hinton και Ronald J. Williams. *Learning Internal Representations by Error Propagation.* Τεχνική Αναφορά με αριθμό, CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE, 1985. Section: Technical Reports.

[33] Yann LeCun, Patrick Haffner, Léon Bottou και Yoshua Bengio. *Object recognition with gradient-based learning. Shape, contour and grouping in computer vision*, σελίδες 319–345. Springer, 1999.

[34] David H Hubel και Torsten N Wiesel. *Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of physiology*, 160(1):106, 1962.

[35] *Neural networks and physical systems with emergent collective computational abilities. | PNAS.*

[36] Sepp Hochreiter και Jürgen Schmidhuber. *Long Short-Term Memory. Neural Computation*, 9(8):1735–1780, 1997. Conference Name: Neural Computation.

[37] M. Gori, G. Monfardini και F. Scarselli. *A new model for learning in graph domains. Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, τόμος 2, σελίδες 729–734, Montreal, Que., Canada, 2005. IEEE.

[38] Y. Lecun, L. Bottou, Y. Bengio και P. Haffner. *Gradient-based learning applied to document recognition. Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[39] Peng Cui, Xiao Wang, Jian Pei και Wenwu Zhu. *A Survey on Network Embedding. arXiv:1711.08752 [cs]*, 2017. arXiv: 1711.08752.

[40] William L. Hamilton, Rex Ying και Jure Leskovec. *Representation Learning on Graphs: Methods and Applications. arXiv:1709.05584 [cs]*, 2018. arXiv: 1709.05584.

[41] Martin Sundermeyer, Ralf Schlüter και Hermann Ney. *LSTM neural networks for language modeling. Interspeech 2012*, σελίδες 194–197. ISCA, 2012.

[42] Xiaolong Wang, Ross Girshick, Abhinav Gupta και Kaiming He. *Non-Local Neural Networks.* σελίδες 7794–7803, 2018.

[43] A. Buades, B. Coll και J. M. Morel. *A non-local algorithm for image denoising. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, τόμος 2, σελίδες 60–65 vol. 2, 2005. ISSN: 1063-6919.

[44] C. Tomasi και R. Manduchi. *Bilateral filtering for gray and color images. Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, σελίδες 839–846, 1998.

[45] Joan Bruna, Wojciech Zaremba, Arthur Szlam και Yann LeCun. *Spectral Networks and Locally Connected Networks on Graphs*, 2014. arXiv:1312.6203 [cs].

[46] *Wavelets on graphs via spectral graph theory - ScienceDirect.*

[47] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik και Ryan P Adams. *Convolutional Networks on Graphs for Learning Molecular Fingerprints. Advances in Neural Information Processing Systems*, τόμος 28. Curran Associates, Inc., 2015.

[48] James Atwood και Don Towsley. *Diffusion-Convolutional Neural Networks. Advances in Neural Information Processing Systems*, τόμος 29. Curran Associates, Inc., 2016.

[49] Chenyi Zhuang και Qiang Ma. *Dual Graph Convolutional Networks for Graph-Based Semi-Supervised Classification. Proceedings of the 2018 World Wide Web Conference*, WWW '18, σελίδες 499–508, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[50] Will Hamilton, Zhitao Ying και Jure Leskovec. *Inductive Representation Learning on Large Graphs. Advances in Neural Information Processing Systems*, τόμος 30. Curran Associates, Inc., 2017.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren και Jian Sun. *Identity Mappings in Deep Residual Networks. Computer Vision – ECCV 2016*Bastian Leibe, Jiri Matas, Nicu Sebe και Max Welling, επιμελητές, Lecture Notes in Computer Science, σελίδες 630–645, Cham, 2016. Springer International Publishing.

[52] Yujia Li, Daniel Tarlow, Marc Brockschmidt και Richard Zemel. *Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493*, 2015.

[53] Michael Kampffmeyer, Yinbo Chen, Xiaodan Liang, Hao Wang, Yujia Zhang και Eric P. Xing. *Rethinking Knowledge Graph Propagation for Zero-Shot Learning. arXiv:1805.11724 [cs]*, 2019. arXiv: 1805.11724.

[54] Yizhou Zhang, Yun Xiong, Xiangnan Kong, Shanshan Li, Jinhong Mi και Yangyong Zhu. *Deep Collective Classification in Heterogeneous Information Networks. Proceedings of the 2018 World Wide Web Conference*, WWW '18, σελίδες 399–408, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[55] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui και Philip S Yu. *Heterogeneous Graph Attention Network. The World Wide Web Conference*, WWW '19, σελίδες 2022–2032, New York, NY, USA, 2019. Association for Computing Machinery.

[56] Yaguang Li, Rose Yu, Cyrus Shahabi και Yan Liu. *Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting*, 2018. arXiv:1707.01926 [cs, stat].

[57] Bing Yu, Haoteng Yin και Zhanxing Zhu. *Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, σελίδες 3634–3640, 2018. arXiv:1709.04875 [cs, stat].

[58] Ashesh Jain, Amir R. Zamir, Silvio Savarese και Ashutosh Saxena. *Structural-RNN: Deep Learning on Spatio-Temporal Graphs.* σελίδες 5308–5317, 2016.

[59] Sijie Yan, Yuanjun Xiong και Dahua Lin. *Spatial temporal graph convolutional networks for skeleton-based action recognition. Thirty-second AAAI conference on artificial intelligence*, 2018.

[60] Robyn Speer, Joshua Chin και Catherine Havasi. *ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[61] Push Singh, Thomas Lin, Erik T. Mueller, Grace Lim, Travell Perkins και Wan Li Zhu. *Open Mind Common Sense: Knowledge Acquisition from the General Public. On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*Robert Meersman και Zahir Tari, επιμελητές, Lecture Notes in Computer Science, σελίδες 1223–1237, Berlin, Heidelberg, 2002. Springer.

[62] L.von Ahn. *Games with a purpose. Computer*, 39(6):92–94, 2006. Conference Name: Computer.

[63] Francis Bond και Ryan Foster. *Linking and Extending an Open Multilingual Wordnet.* σελίδα 11.

[64] *WordNet: An Electronic Lexical Database - George A. Miller - Google Books.*

[65] James Breen. *JMdict: a Japanese-multilingual dictionary. Proceedings of the Workshop on Multilingual Linguistic Ressources - MLR '04*, σελίδα 71, Geneva, Switzerland, 2004. Association for Computational Linguistics.

[66] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak και Zachary Ives. *DBpedia: A Nucleus for a Web of Open Data. The Semantic Web*Karl Aberer, Key Sun Choi, Natasha Noy, Dean Allemang, Kyung Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber και Philippe Cudré-Mauroux, επιμελητές, Lecture Notes in Computer Science, σελίδες 722–735, Berlin, Heidelberg, 2007. Springer.

[67] Oliver Lemon, Kallirroi Georgila, James Henderson και Matthew Stuttle. *An ISU Dialogue System Exhibiting Reinforcement Learning of Dialogue Policies: Generic Slot-filling in the TALK In-car System.* σελίδα 4.

[68] Karen Sparck Jones. *Natural Language Processing: A Historical Review. Current Issues in Computational Linguistics: In Honour of Don Walker*Antonio Zampolli, Nicoletta Calzolari και Martha Palmer, επιμελητές, Linguistica Computazionale, σελίδες 3–16. Springer Netherlands, Dordrecht, 1994.

[69] Yoshua Bengio, Réjean Ducharme και Pascal Vincent. *A Neural Probabilistic Language Model. Advances in Neural Information Processing Systems*, τόμος 13. MIT Press, 2000.

[70] Jeffrey Pennington, Richard Socher και Christopher Manning. *Glove: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language*

*Processing (EMNLP)*, σελίδες 1532–1543, Doha, Qatar, 2014. Association for Computational Linguistics.

[71] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee και Luke Zettlemoyer. *Deep contextualized word representations*, 2018. Number: arXiv:1802.05365 arXiv:1802.05365 [cs].

[72] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh και Dhruv Batra. *Visual dialog. Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 326–335, 2017.

[73] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár και C Lawrence Zitnick. *Microsoft coco: Common objects in context. European conference on computer vision*, σελίδες 740–755. Springer, 2014.

[74] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick και Devi Parikh. *Vqa: Visual question answering. Proceedings of the IEEE international conference on computer vision*, σελίδες 2425–2433, 2015.

[75] Yuke Zhu, Oliver Groth, Michael Bernstein και Li Fei-Fei. *Visual7w: Grounded question answering in images. Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 4995–5004, 2016.

[76] Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang και Wei Xu. *Are you talking to a machine? dataset and methods for multilingual image question. Advances in neural information processing systems*, 28, 2015.

[77] Harsh Agrawal, Arjun Chandrasekaran, Dhruv Batra, Devi Parikh και Mohit Bansal. *Sort story: Sorting jumbled images and captions into stories. arXiv preprint arXiv:1606.07493*, 2016.

[78] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra και Devi Parikh. *Making the v in vqa matter: Elevating the role of image understanding in visual question answering. Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 6904–6913, 2017.

[79] Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra και Devi Parikh. *Yin and yang: Balancing and answering binary visual questions. Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 5014–5022, 2016.

[80] Bo Dai, Yuqi Zhang και Dahua Lin. *Detecting Visual Relationships With Deep Relational Networks.* σελίδες 3076–3086, 2017.

[81] Yulei Niu, Hanwang Zhang, Manli Zhang, Jianhong Zhang, Zhiwu Lu και Ji Rong Wen. *Recursive visual attention in visual dialog. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, σελίδες 6679–6688, 2019.

[82] Zilong Zheng, Wenguan Wang, Siyuan Qi και Song Chun Zhu. *Reasoning visual dialogs with structural and partial observations. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, σελίδες 6669–6678, 2019.

[83] Shubham Agarwal, Trung Bui, Joon Young Lee, Ioannis Konstas και Verena Rieser. *History for Visual Dialog: Do we really need it? arXiv preprint arXiv:2005.07493*, 2020.

[84] Xiaoze Jiang, Jing Yu, Zengchang Qin, Yingying Zhuang, Xingxing Zhang, Yue Hu και Qi Wu. *Dualvd: An adaptive dual encoding model for deep visual understanding in visual dialogue. Proceedings of the AAAI conference on artificial intelligence*, τόμος 34, σελίδες 11125–11132, 2020.

[85] Feilong Chen, Fandong Meng, Jiaming Xu, Peng Li, Bo Xu και Jie Zhou. *Dmrm: A dual-channel multi-hop reasoning model for visual dialog. Proceedings of the AAAI Conference on Artificial Intelligence*, τόμος 34, σελίδες 7504–7511, 2020.

[86] Xiaoze Jiang, Jing Yu, Yajing Sun, Zengchang Qin, Zihao Zhu, Yue Hu και Qi Wu. *DAM: Deliberation, abandon and memory networks for generating detailed and non-repetitive responses in visual dialogue. arXiv preprint arXiv:2007.03310*, 2020.

[87] Vishvak Murahari, Dhruv Batra, Devi Parikh και Abhishek Das. *Large-scale pretraining for visual dialog: A simple state-of-the-art baseline. European Conference on Computer Vision*, σελίδες 336–352. Springer, 2020.

[88] Yue Wang, Shafiq Joty, Michael R Lyu, Irwin King, Caiming Xiong και Steven CH Hoi. *Vd-bert: A unified vision and dialog transformer with bert. arXiv preprint arXiv:2004.13278*, 2020.

[89] Ji Zhang, Yannis Kalantidis, Marcus Rohrbach, Manohar Paluri, Ahmed Elgammal και Mohamed Elhoseiny. *Large-Scale Visual Relationship Understanding. Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9185–9194, 2019. Number: 01.

[90] Zhen Wang, Jianwen Zhang, Jianlin Feng και Zheng Chen. *Knowledge Graph Embedding by Translating on Hyperplanes. Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1), 2014. Number: 1.

[91] Zhe Gan, Yu Cheng, Ahmed El Kholy, Linjie Li, Jingjing Liu και Jianfeng Gao. *Multi-step Reasoning via Recurrent Dual Attention for Visual Dialog*, 2019. arXiv:1902.00579 [cs].

[92] Diederik P. Kingma και Jimmy Ba. *Adam: A Method for Stochastic Optimization*, 2017. arXiv:1412.6980 [cs].

# List of Abbreviations

| | |
|---|---|
| AI | Artificial Intelligence |
| BERT | Bidirectional Encoder Representations from Transformers |
| DL | Deep Learning |
| ELMo | Embeddings from Language Models |
| GAT | Graph Attention Network |
| GCN | Graph Convolutional Network |
| GN | Graph Network |
| GNN | Graph Neural Network |
| GRU | Gated Recurrent Unit |
| LSTM | Long short-term memory |
| ML | Machine Learning |
| NMT | Neural Machine Translation |
| MLP | Multilayer Perceptron |
| MNLI | The Multi-Genre Natural Language Inference |
| MRPC | The Microsoft Research Paraphrase Corpus |
| NAS | Neural Architecture Search |
| NLI | Natural Language Inference |
| NLG | Natural Language Generation |
| NLP | Natural Language Processing |
| NLU | Natural Language Understanding |
| NNLM | Neural Network Language Model |
| QA | Question Answering |
| QNLI | Question-answering Natural Language Inference |
| R-GCN | Relational Graph Convolutional Network |
| RNN | Recurrent Neural Network |
| RTE | Recognizing Textual Entailment |
| VD | Visual Dialog |
| WNLI | Winograd Natural Language Inference |
| XAI | Explainable Artificial Intelligence |

Chapter 7

# Appendix

## 7.1 Implementation Details

To extract the features for the Visual Knowledge Graph, we utilize Faster-RCNN [9] with the ResNet-101 to pick up top 36 object regions (i.e. N = 36) and produce the 2048-dimension region features. The maximum sentence length of the dialogue history and the current question are set to 20. The hidden state size of LSTM blocks is all set to 512. This is the dimension of each node in the Text Knowledge Graph. The dimension of each edge in the graph is all set to 512. For the External Knowledge Graph, we use the Numberbatch embeddings, with dimension equal to 300. The maximum number of original nodes int the External Knowledge Graph is set to 45, while each original node can have up to 45 neighbours. We use Adam [92] optimizer to train our model, utilizing cross entropy loss. As in the baseline, we first conduct warm-up strategy, which trains the model with initial learning rate 1e-3 and warm-up factor 0.2 for 2 epochs and then utilizes cosine annealing learning strategy with initial learning rate 1e-3 and final learning rate 3.4e-4 for the rest of epochs. The mini-batch size is 4, the drop ratio of the External Knowledge Encoding is set to 0.1, while for the rest of the model to 0.5.

## 7.2 Preprocessing

### 7.2.1 Dataset preprocessing

As stated in the baseline description, the initialization both of the vision and the text nodes is done with pre-extracted embeddings. Specifically, for the vision nodes, Faster-RCNN [9] was utilized with the ResNet-101 to pick up top 36 object regions (i.e. N = 36) and produce the 2048-dimension region features. In addition, for the edges of the vision graph the visual relationships were extracted by a visual relationship encoder [89]. For the text graph, each round of each dialog was properly preprocessed, as in every Natural Language Processing problem. After selecting only the words that occurred at least 5 times in the whole dataset, the maximum length of the dialogue history and the one of the current question was set to 20, in order to ensure fixed sized inputs.

### 7.2.2 Preprocess of ConceptNet

Fetching the relevant external information is crucial, in order to achieve better results. To avoid querying the whole Concept-Net graph during training, a preprocessing of the Concept-Net is necessary, resulting in a subgraph of the knowledge graph, for each round of each dialog. Specifically, we take into account all of the concepts present in the caption, the dialog history and the current question of each round. The concepts that correspond to the objects that are visible in the image could be also taken into account, but experiments showed that every object detected by the Faster-RCNN was already present as a concept, as it was mentioned in the caption.

The subgraph for each round of each dialog is populated by the original nodes that correspond to concepts already present in the dialog history or in the current question, as well as by nodes that correspond to concepts related to the original ones. Formally, we define the External Knowledge Graph as the directed graph:

$$G = (V^o \cup V^e \subseteq V^{cpnet}, E) \tag{7.1}$$

where $V^{cpnet}$ is the set of nodes of the whole ConceptNet Knowledge Graph, $V^o$ is the set of the original nodes and $V^e$ the set of the extra nodes. The set of the edges can then be defined as:

$$E = \{e_{ij} = (v_i, v_j) | (v_i, v_j) \epsilon V^2 \wedge r(v_i, v_j) \epsilon R\} \tag{7.2}$$

where $R$ is the set of relation types that we take into account and $r$ a function that assigns each edge to its type. The set of all the relation types that were taken into account for the current work are presented in Table 7.2.

One approach to select the extra nodes that will be added to $V^e$ is to choose the ones that are present in a path of maximum length $K \geq 1$ between two original nodes, regardless the relation type they are involved to. A simple visualization of such paths is shown in Fig. 7.1. We depict the original nodes as the blue ones. Paths with length greater than $K$ will not be accepted, hence the red nodes will not be added to $V^e$.

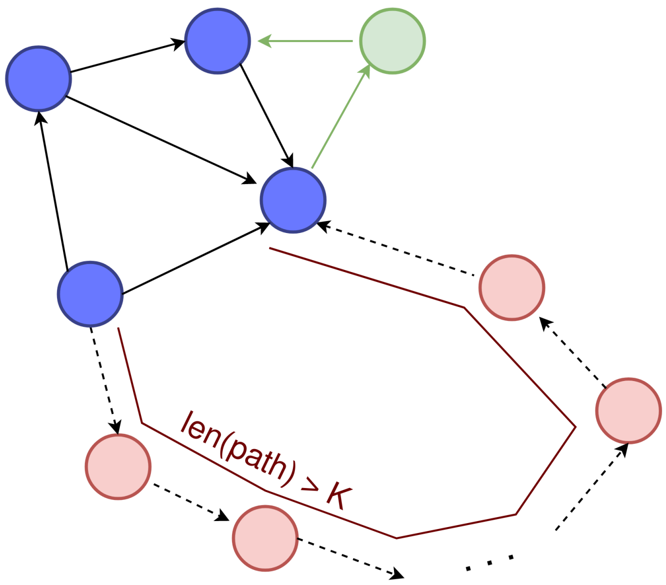


**Figure 7.1.** *Blue nodes represent the original nodes, green the nodes that are selected for the $V^e$ set, while in the red ones the nodes that are not.*

Due to computational reasons, we limited in taking into account only paths of length equal to two, $K = 2$. This implies that $V^e$ will be populated with nodes that are the common neighbours of the original nodes, regardless the relation type. A simple visualization is once again shown in Fig. 7.2.

The algorithm implemented for the preprocessing of ConceptNet is shown below:

For each round $r$ of the dialog:

1. Initialize the sets $V_o^r$ and $V_e^r$ as empty sets.

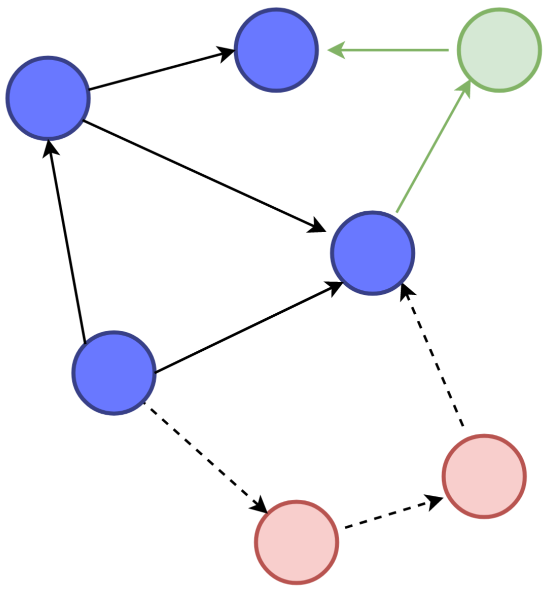


**Figure 7.2.** *Only the green node will be added to $V^e$ and the red ones will be discarded.*

2. Pair dialog entities (tokens) with concepts of the ConceptNet. The dialog entities are gathered from the current question, as well as the dialog history. Add these concepts to the set of the original nodes $V^r_o$.

3. If in the current round the number of concepts exceeds a fixed limit keep only last $max\_nodes$ nodes, which means the concepts that were more recently referenced in the dialog.

4. For each pair of concepts the find paths, whose length is shorter or equel to 2 and add the in between extra nodes that were visited to the set $V^r_e$.

5. For each node in $V^r_o$ and $V^r_e$, initialize $17 * 2 + 1$ empty lists of neighbours, each list corresponding to a different relation type and add the node to the last relation type (self relation)

6. Using TransE [90], score all the edges and select those with score above a fixed threshold. The score is computed as the cosine similarity between the TransE embeddings of the examined nodes.

7. For every node:
   (a) Sort all of its neighbours, based on the TransE score.
   (b) Keep only the first $max\_edges$ edges and the involved nodes.

8. For every selected node $v$:
   (a) For every node's neighbour $u \epsilon N_v$:
      i. If node is original, $v \epsilon V^r_o$, and edge limit for the specific relation type is not exceeded, add neighbour $u$ to node's list of neighbours.
      ii. Else if the neighbour node is an original node, $u \epsilon V^r_o$, and edge limit for the inverse relation type is not exceeded, add node to neigbour's list of neighbours (inverse relation).

9. While creating the above adj_list for each round, a list of the mentioned concepts is being kept. By iterating all the dialogs' last round we get a set of all the mentioned concepts

in VisDial dataset. This set is used to extract a *concepts vocabulary* to be used for the numberbatch embeddings.

By the end of the above procedure there will be a graph for every round of all the dialogs. The graph will have a total of **17 * 2 + 1** types of relations (listed below). Each node will be represented by the Numberbatch embedding, which will be used for the initialization of the node embedding in the GNN. Numberbatch embeddings are built using an ensemble that combines data from ConceptNet, word2vec, GloVe, and OpenSubtitles 2016.

As mentioned above, we take into account 17 relation types, the corresponding inversed ones and a self relation. These relation types are listed below, in table 7.2:

| All relations | Common relations |
|---|---|
| antonym | antonym |
| atlocation | atlocation |
| capableof | capableof |
| causes | causes |
| createdby | isa |
| isa | desires |
| desires | hassubevent |
| hassubevent | partof |
| partof | hasproperty |
| hascontext | relatedto |
| hasproperty | usedfor |
| madeof | inv_antonym |
| notcapableof | inv_atlocation |
| notdesires | inv_capableof |
| receivesaction | inv_causes |
| relatedto | inv_isa |
| usedfor | inv_desires |
| inv_antonym | inv_hassubevent |
| inv_atlocation | inv_partof |
| inv_capableof | inv_hasproperty |
| inv_causes | inv_relatedto |
| inv_createdby | inv_usedfor |
| inv_isa | self_relation |
| inv_desires | |
| inv_hassubevent | |
| inv_partof | |
| inv_hascontext | |
| inv_hasproperty | |
| inv_madeof | |
| inv_notcapableof | |
| inv_notdesires | |
| inv_receivesaction | |
| inv_relatedto | |
| inv_usedfor | |
| selfrelation | |

**Table 7.2.** *The relation types used for creating the subgraphs of ConceptNet.*

### Preprocessing results analysis

For each split of the dataset, a barplot showing the number of relations present in each dialog is shown in diagrams 7.3, 7.4 and 7.5.

We can see that the relation *related_to* is the most common one. This is expected, as every node in the Concept-Net knowledge graph is very likely to be connected with many more nodes through the *related_to* relation, than any other relation mentioned above. This imbalance in the frequency of each relation type may lead to less impressive results, if we treat each type independently and in the same way. As [27] suggested the use of common weights for all different relation types, through basis decomposition of the weights and this is the method that we used, as described in Section 5.5.2

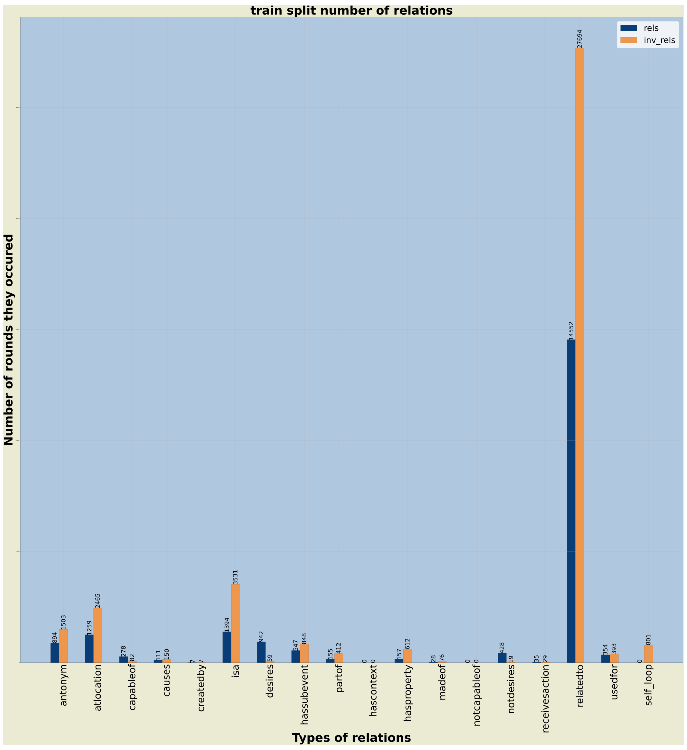


**Figure 7.3.** *Frequency of relations in train split.*

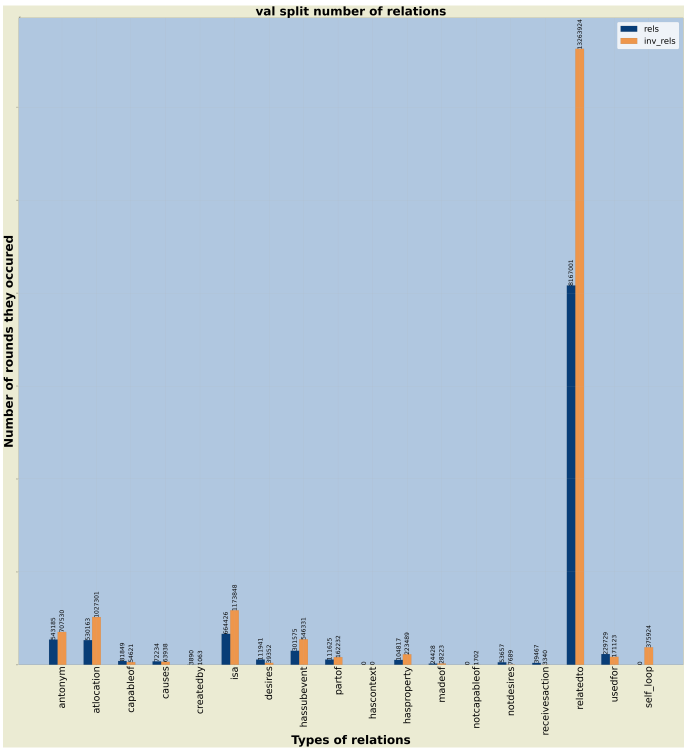


**Figure 7.4.** *Frequency of relations in val split.*

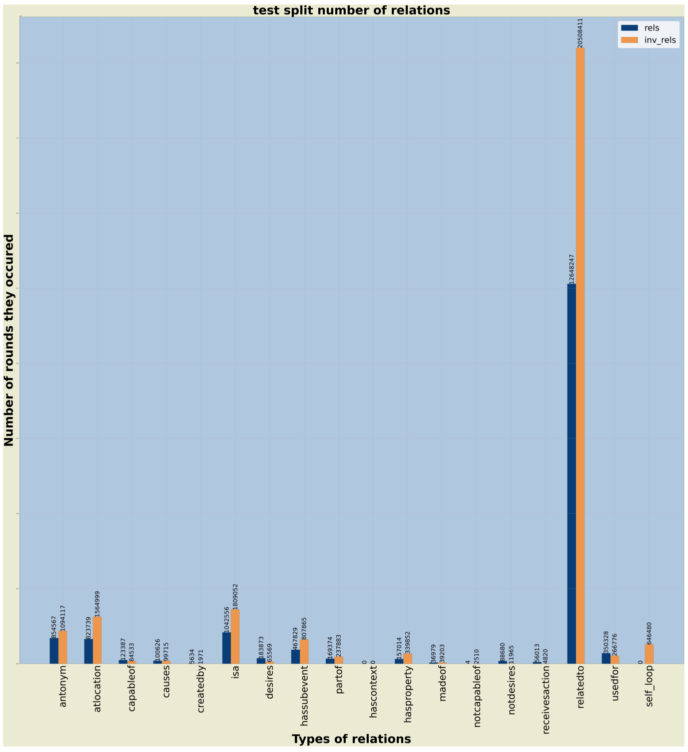


**Figure 7.5.** *Frequency of relations in test split.*