CS 4644 / 7643-A: LECTURE 6
DANFEI XU

Topic: Course Project

Today’s Lecture

Topics:

- What is the course project about?
- How to pick a project?
- What is computer science research?
- How to develop a new idea?
- How to read a paper?

What is the course project about?

What we want you to have:

- Hands-on experience with Deep Learning
- In-depth understanding of a specific problem / algorithm / dataset / library / network architecture
- Experience with teamwork
- (Optional) Opportunity to do research
- Have fun!

What is the course project about?

Completed in group of 2-4 people.

- Project expectations are higher for groups with more people

In general, three tracks of works (sometimes combined):

- **Applications**: If you have a specific background or interest (e.g., biology, physics, medical). We'd love to see you apply Deep Learning to your particular domain of interest.
- **Experimental analysis**: do an in-depth study on a class of algorithms / models. Why does the model work so well on this problem?
- **Research**: build on top of an existing algorithm / model and try to improve on it.

What’s the scope of project topics?

Any problem that makes (heavy) use of deep neural networks.

Examples:

- Computer Vision: Classification, Detection, Reconstruction, …
- Natural Language Processing: Language Modeling, Translation, Story-telling, …
- Graphics: Rendering, Animation, Simulation, …
- Robotics: Grasping, Control, Planning, Driving, RL, …
- Bio/Medical: Medical Imaging, Pathology, Gene Data Analysis, Protein Folding, …
- Music / audio: Synthesis, Classification, Style Transfer, …

…

How will my project be graded?

- Project proposal (1%): Sep 26th
  - Mainly for us to provide feedback to your project ideas.
  - *What*: What is the problem you want to invest?
  - *So what*: Why is it interesting? Why should others care?
  - *Now what:* What is your proposed course of action?
    - What readings will you examine to provide context and background?
    - What data will you use? If you are collecting new data, how would you do it?
    - What models or implementation will you use? How do you plan to improve or modify the existing method?
    - How will you evaluate your results quantitatively (performance measure) and qualitatively (visualization)? What kind of analysis do you expect to run?

How will my project be graded?

- Milestone presentation (5%): due date TBD
  - Present your project to the teaching team (individual appointment or office hour)
  - 5-minutes presentation
  - Motivation, problem setup, related works, proposed method, preliminary results, plans moving forward.

How will my project be graded?

- Milestone report (5%): due date TBD
  - **Problem**: Provide a concrete problem statement and formulation (equations, input / output)
  - **Method**: You should have already implemented a basic version of your algorithm / model and show that it's working (for your problem)
  - **Dataset**: Finished data preprocessing (if applicable). If you need to collect new data, you should have already done that.
  - **Experiment**: If you want to improve on a model, you should already have the results on the baseline (the method you are based off of).

How will my project be graded?

- Final Project Report (20%): Dec 4
  - Practically speaking, you are writing a short paper
    - Introduction
    - Related Works
    - Method
    - Dataset
    - Methods
    - Experiments
    - Discussion & Conclusions

How will my project be graded?

- Poster Session (5%): Dec 5th
  - Open to GT (we will advertise it)
  - Present your project to your classmates / friends / professors / …
  - **Keep your target audience in mind**. You are not talking to domain experts!
  - In person, Klaus Atrium.
  - TAs will evaluate your poster on the spot

Does my project meet expectations?

Rule of thumb: **Did I put enough effort into my project?**

- Everyone has a different background. We do not expect you to do the same kind of projects as students who are already doing DL research.
- We grade based on effort. We most likely can tell how much effort you put into a project by looking at your project final report / poster.

Does my project meet expectations?

This **does not** mean:

- Your project has to be strictly novel to get a good grade (although encouraged)
- Your method has to beat the state-of-the-art method to get a good grade

This **does** mean:

- You need to put significant effort into your investigation, which may involve trying many different approaches
- When you do analysis, ask yourself: are you explaining and understanding your results, or merely stating them?

Adapted from Stanford CS231n Project Guideline

What is a bad project?

- **Too ambitious:**
  - Need to spend weeks / months cleaning or collecting data
  - Need $$$ to label data on Amazon Mechanical Turk
  - Do something completely new / go against findings in existing research (ignore this if you know what you are doing).
- **Not ambitious enough:**
  - Apply model (existing implementation) X to dataset Y with minimal change.
    - E.g., Clone a repo and do minimal stitching to make it work for a Kaggle competition.
- An idea that you are ***not*** interested in, but you are doing it because it’s easy / convenient / other people have done similar things.
  - You will be far more motivated if you are invested in what you are doing.

How do I pick a (good) project?

- What do you care about? Better medical AI? Better Autonomous Driving? WebAgent? Visual Art?
- Does it help you learn new things that will be useful later
- Practical Considerations:
  - Data: Is there dataset / simulation environment for your problem? Is it accessible?
  - Code & framework: Do I have to implement everything from scratch? Or can I base it off of something?
  - What’s the simplest way (baseline) to approach the problem?

A note on using foundation models in your project

- It is ok to have pretrained foundation models (e.g., LLMs, CLIP) as a component of your project
- But you will have to do more than just building a conventional program around a foundation model.
- Examples: It is ok do a project that involves …
  - Training / tuning a small Language Model
  - Investigate the most effective LORA for SegmentAnything
  - Develop a new vision-language model by fusing existing LLM / VLM
- Examples: It is NOT ok to do a project that …
  - builds a chatbot interface based on existing LLMs
  - builds a front-end wrapper around StableDiffusion
- If you are unsure, talk to the teaching team

FAQ

Q: Can I change my project after proposal, ***before*** the milestone

A: Yes, the proposal is to make sure you have a plausible project direction. If you need to change the direction, we understand.

Q: Can I change my project ***after*** the milestone?

A: No unless you can convince us. At this point in the course, there will be little time to put together a sufficient project.

Q: How do I get help for my project?

A: Come to office hours! We will post a list of TA expertise.

Today’s Lecture

- What is the course project about?
- How to pick a project?
- What is computer science research?
- How to develop a new idea?
- How to read a paper?

Computer Science Research

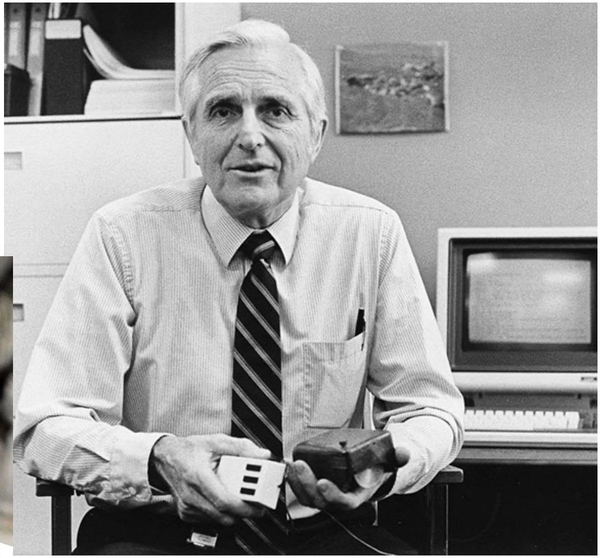
What is the goal of research?

Why has it driven major innovations in computing?

What separates research from engineering development?

Your project doesn't need to be strictly research-based, but you should approach it with a research mindset.

A Tale of Three Turing Awards

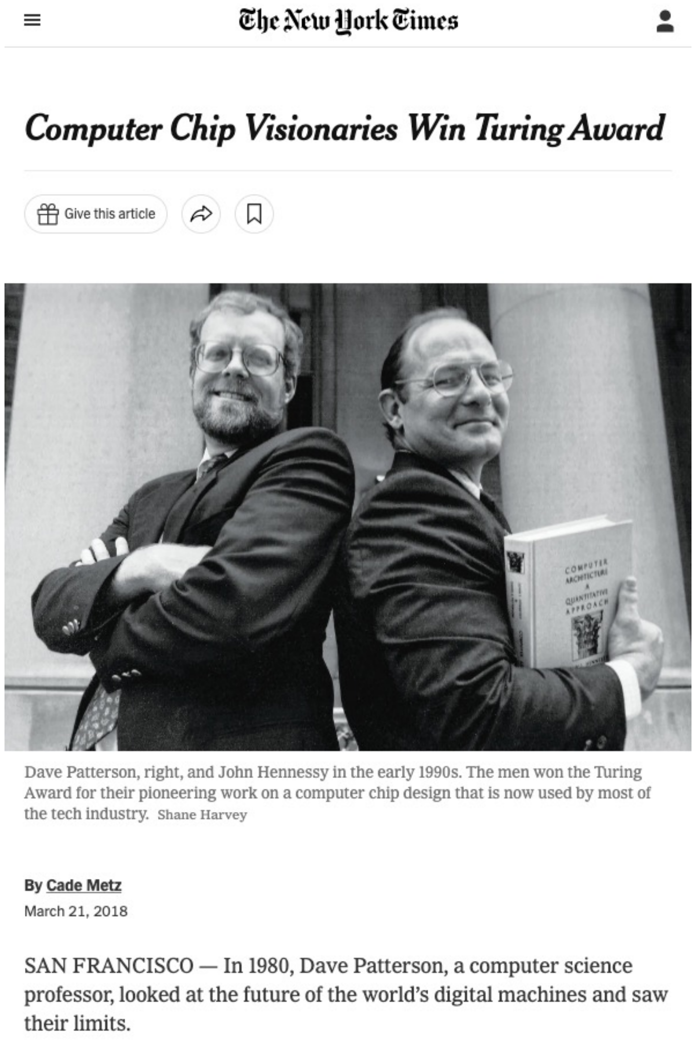


Hennessy and Patterson: RISC

Computer architecture was increasing in complexity, in order to enable more and more advanced computation.

Everyone thought that *increasingly powerful processors needed increasingly complicated instruction* sets to take advantage of them.

Adapted from Stanford CS197



The New York Times

Computer Chip Visionaries Win Turing Award

Give this article

Dave Patterson, right, and John Hennessy in the early 1990s. The men won the Turing Award for their pioneering work on a computer chip design that is now used by most of the tech industry. Shane Harvey

By Cade Metz

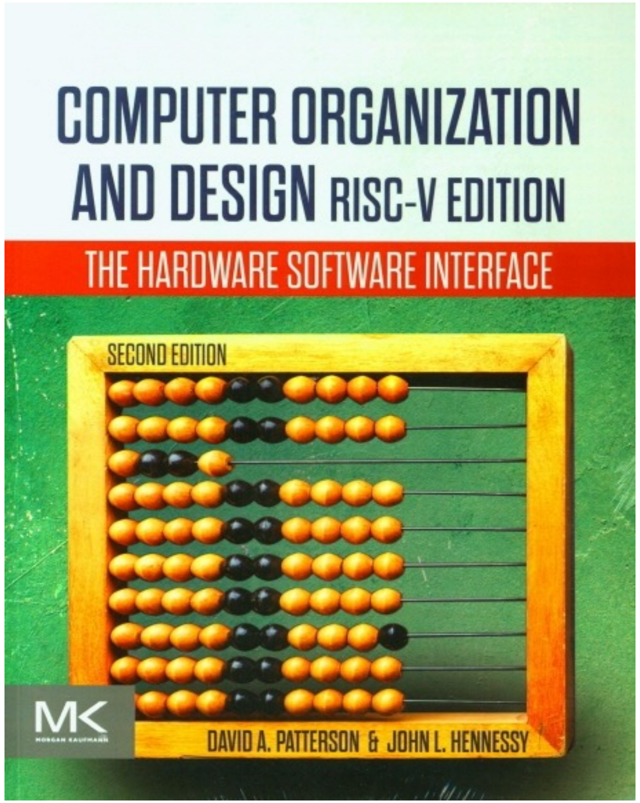
March 21, 2018

SAN FRANCISCO — In 1980, Dave Patterson, a computer science professor, looked at the future of the world's digital machines and saw their limits.

Hennessy and Patterson: RISC

"No, let's do it this way instead:" have a very simple instruction set. That way you can compare performance, optimize, and prevent errors.

This became known as Reduced Instruction Set Computer (RISC). Today, more than 99 percent of all new chips use the RISC architecture they developed.

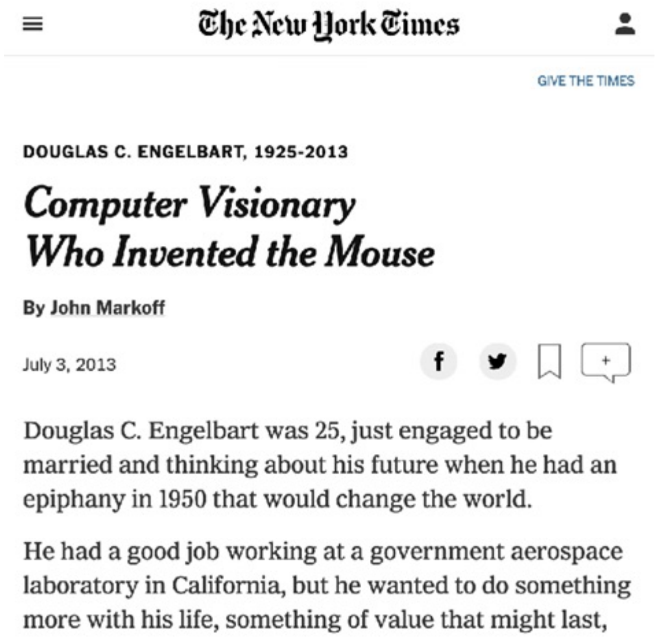

COMPUTER ORGANIZATION AND DESIGN RISC-V EDITION
THE HARDWARE SOFTWARE INTERFACE
SECOND EDITION
MK
DAVID A. PATTERSON & JOHN L. HENNESSY

Adapted from Stanford CS197

Engelbart: interactive computing

When computers originated, they were used for, well, computing: calculating mathematical functions.

This meant that computers were seen as most appropriate for slow, batch interaction, shared by entire teams.



The New York Times

GIVE THE TIMES

DOUGLAS C. ENGELBART, 1925-2013

Computer Visionary Who Invented the Mouse

By John Markoff

July 3, 2013

Douglas C. Engelbart was 25, just engaged to be married and thinking about his future when he had an epiphany in 1950 that would change the world.

He had a good job working at a government aerospace laboratory in California, but he wanted to do something more with his life, something of value that might last,

Adapted from Stanford CS197

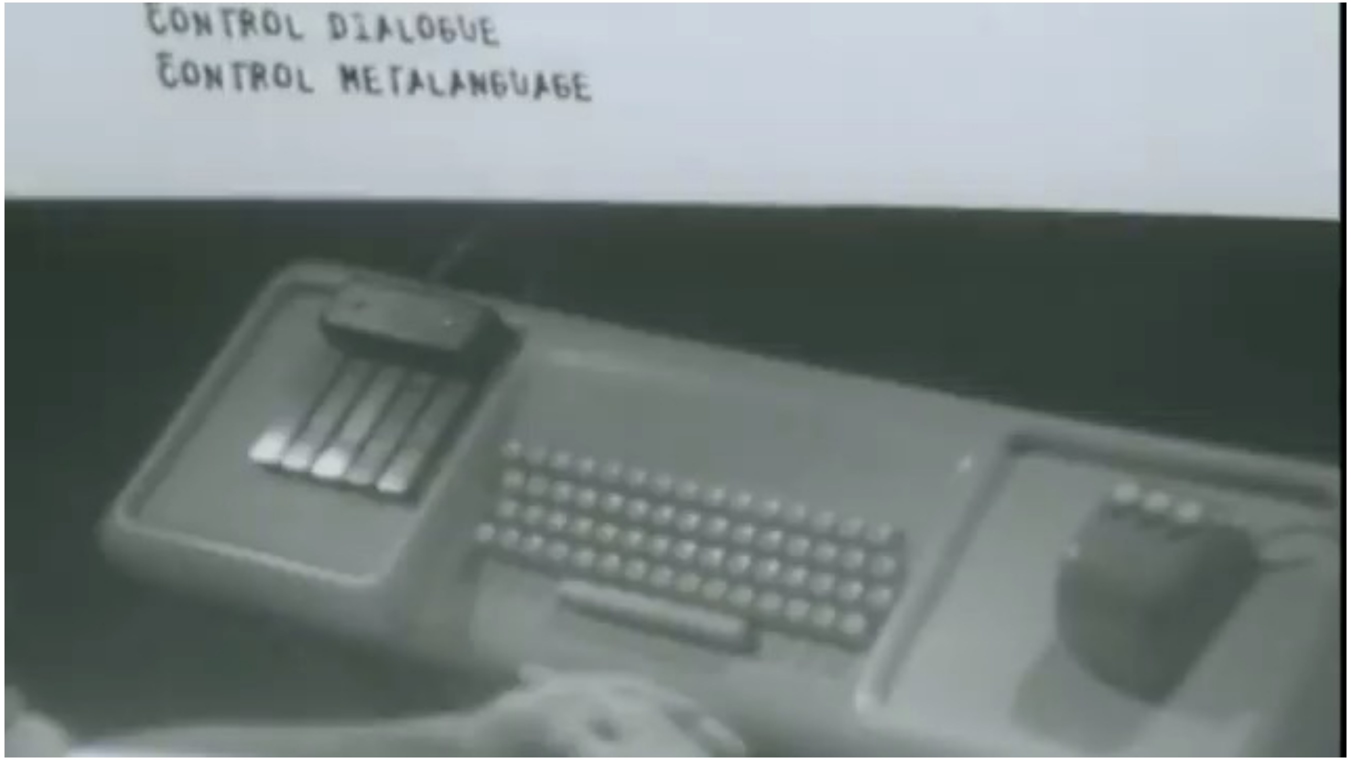
Engelbart: interactive computing

"No, let's do it this way instead:" computing should be used as a tool for thought. We must move from batch-style computing to interactive computing.

His result was the "Mother of All Demos": mouse, hypertext, bitmapped screens, collaborative software, and more.

This led to Xerox Star. Steve Jobs saw it, was wow'ed, and infused the ideas into the Mac.

Engelbart: interactive computing

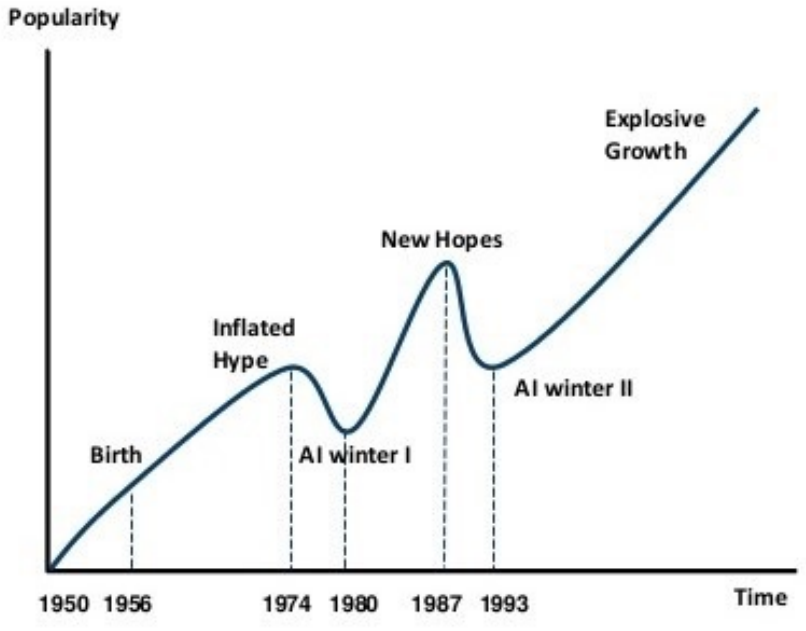


https://www.youtube.com/watch?v=JQ8ZiT1sn88

LeCun, Hinton, Bengio: deep learning

The idea of neural networks had been around for fifty years, but unsuccessful. Major AI figures had trashed it, even proving that early versions had very limited expressiveness.

Instead, machine learning was based on other models, for example the support vector machine and graphical models. Neural networks did not perform well.

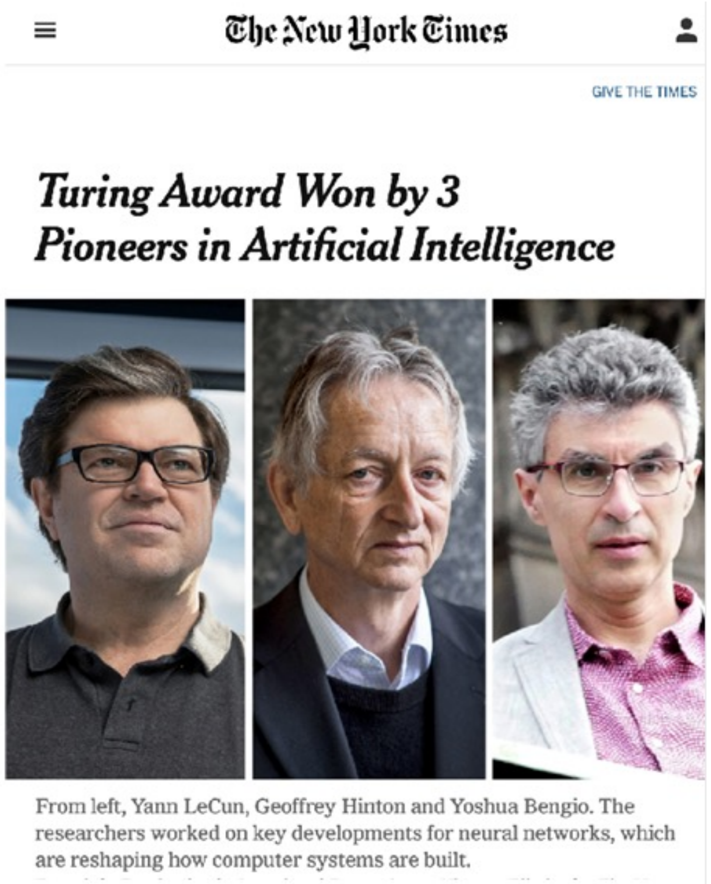


Adapted from Stanford CS197

LeCun, Hinton, Bengio: deep learning

"No, let's do it this way instead:" these networks learn extremely complex functions, so they need much more data than existing machine learning approaches, massively parallel computing to train, and algorithms to enable them to learn more effectively.

Around 2010, these models began smashing records in speech and image recognition. They are now foundational to ML and AI in general.



The New York Times

GIVE THE TIMES

Turing Award Won by 3 Pioneers in Artificial Intelligence

From left, Yann LeCun, Geoffrey Hinton and Yoshua Bengio. The researchers worked on key developments for neural networks, which are reshaping how computer systems are built.

Not all research wins Turing Awards. But…

It all follows the same formula:

An implicit assumption: Industry and other researchers all thought one way about a problem

“No, let’s do it this way instead:” The researcher offered a new perspective that nobody had ever considered or made feasible before. They proved out their idea as the better approach.

Adapted from Stanford CS197

What is research?

Research introduces a fundamental **new idea** into the world.

These ideas did not exist in any mature or well-articulated way before their creators developed them.

If the idea is already in the world, for example published by someone else, it is not considered novel, and thus not research.

Adapted from Stanford CS197

How to develop a novel idea?

Novel ideas rarely come out of a vacuum

They're much more often pivoted off of today's work:

- A realization that an idea has been applied in domains like X and needs to be rethought in domains like ~X
- A recognition that others have tried this technique in users of context A, or data of up to size N, but ~A or >>N breaks the technique, or enables new behaviors
- Some constraint that exists but shouldn't, or visa versa

The "bit flip” method: invert an assumption

bit flip: an inversion of an assumption that the world has about how the world is supposed to work.

Not trying to flip *any* bit. Guided by existing research, logic, and good intuition.

The "bit flip" method: invert an assumption

bit flip: an inversion of an assumption that the world has about how the world is supposed to work.

Recipe for a bit flip:

1) Define the bit: *articulate* an assumption, often left implicit in prior work
2) Introduce the flip: argue for an *alternative* to that assumption / "No, let's do it this way instead"
3) Justify the flip: *so what*? Why does flipping this bit matter?

Bit	Flip	Project
We need complicated instruction sets to accommodate powerful computer processors.	Simple instruction sets are better since they let you compare performance, optimize, and prevent errors.	RISC architecture
Computing was just for numerical calculations: slow, done in batches, and for teams.	Computing should be interactive, individual, and support thought.	Mother of all demos
Neural networks exist, but don't perform very well and aren't accurate.	We need more data & compute, and different algorithms for this to work.	Deep learning

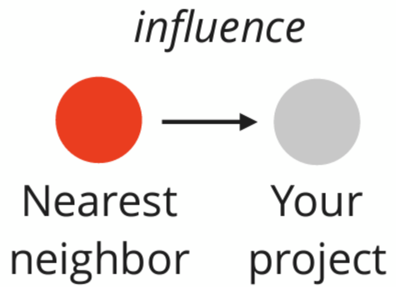
Bit	Flip	Project
Activity tracking requires custom hardware.	Activity tracking requires just a standard cell phone.	
NLP machine learning models should read sentences word by word (Recurrent Neural Nets)	Models should consume the entire sentence at once to make long-range dependency learning easier	Transformer Networks
…	…	Your awesome project

Single paper bit slip

Papers / ideas are like points in high-dimensional space …

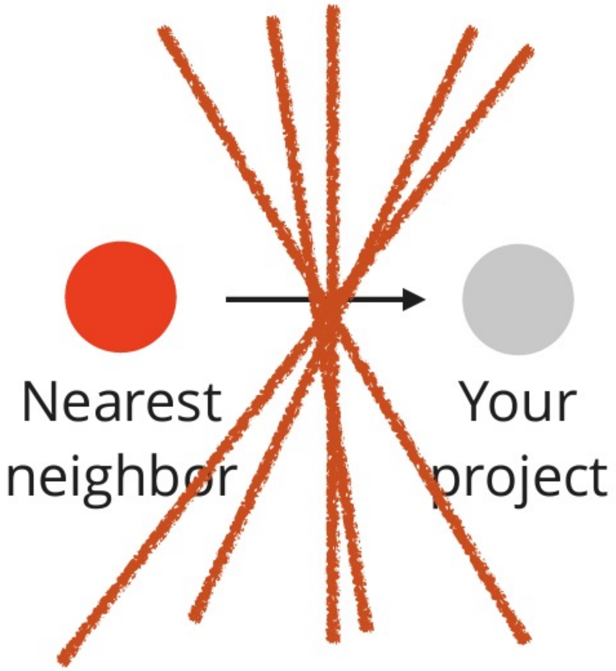
Find a paper that is adjacent to your idea. Think of this as your nearest neighbor paper.

Your project will be some sort of delta off of that paper. What assumption or limitation did it have, that you're erasing?



Adapted from Stanford CS197

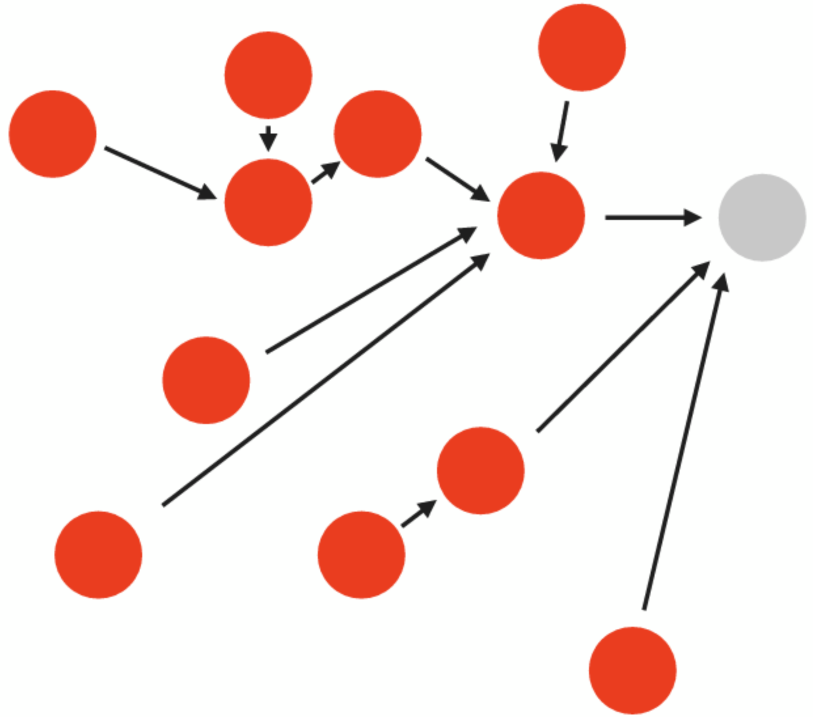
Single paper bit slip



Each separating line is a possible bit flip.

Which one should you go for?

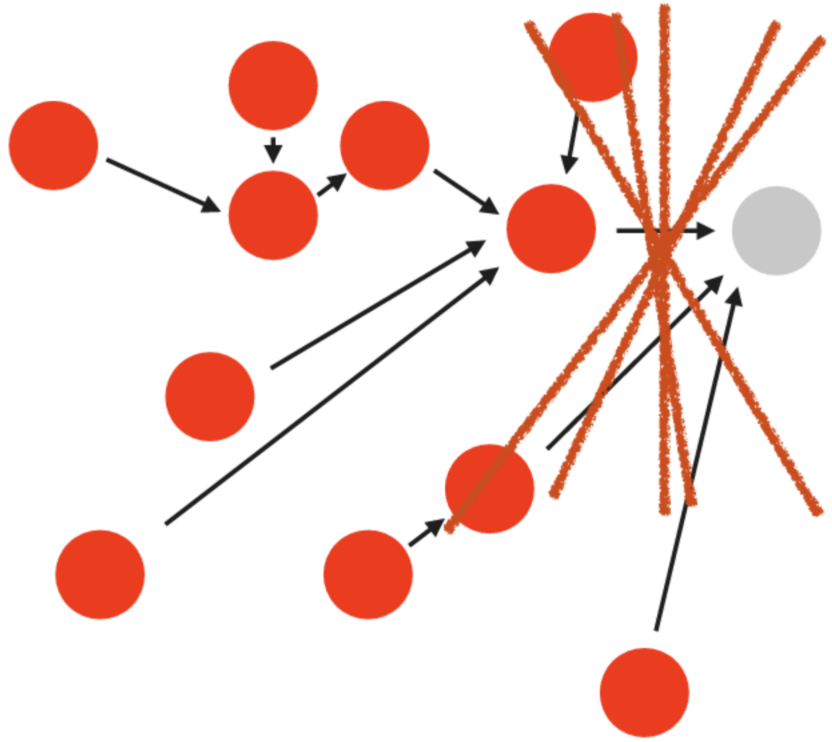
Literature bit flip



The broader an understanding you have of the literature and the assumptions underneath each paper and their commonalities and differences, the more effectively you can pick the right bit flip.

Adapted from Stanford CS197

Literature bit flip



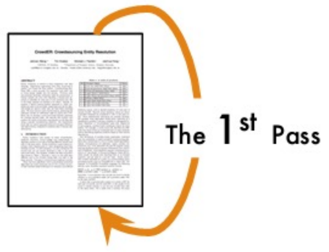
The broader an understanding you have of the literature and the assumptions underneath each paper and their commonalities and differences, the more effectively you can pick the right bit flip.

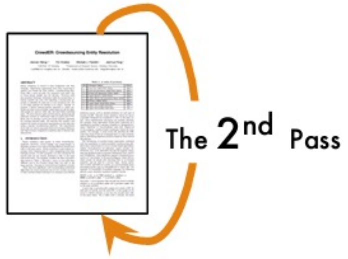
Adapted from Stanford CS197

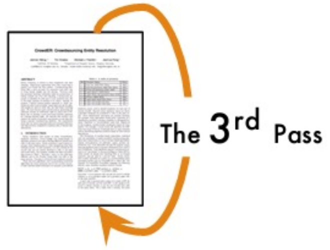
How to read a paper

The "three-pass" approach [1]

- first pass: a quick scan
- second pass: with greater care, but ignore the details
- third pass: re-implementing the paper

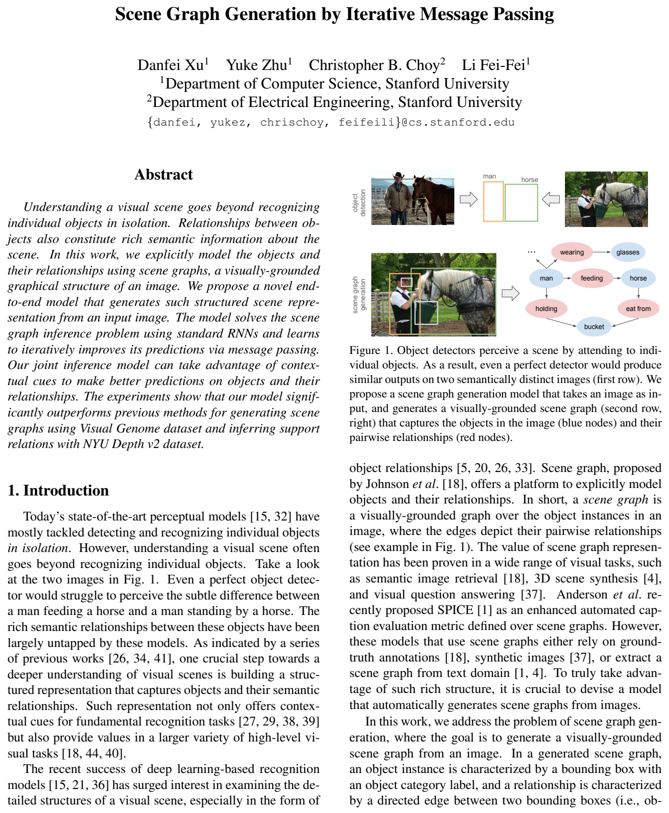






[1] S. Keshav. How to read a paper? http://blizzard.cs.uwaterloo.ca/keshav/home/Papers/data/07/paper-reading.pdf

An anatomy of a (computer vision) paper



Scene Graph Generation by Iterative Message Passing

Danfei Xu[1] Yuke Zhu[1] Christopher B. Choy[2] Li Fei-Fei[1]
[1]Department of Computer Science, Stanford University
[2]Department of Electrical Engineering, Stanford University
{danfei, yukez, chrischoy, feifeili}@cs.stanford.edu

Abstract

Understanding a visual scene goes beyond recognizing individual objects in isolation. Relationships between objects also constitute rich semantic information about the scene. In this work, we explicitly model the objects and their relationships using scene graphs, a visually-grounded graphical structure of an image. We propose a novel end-to-end model that generates such structured scene representation from an input image. The model solves the scene graph inference problem using standard RNNs and learns to iteratively improves its predictions via message passing. Our joint inference model can take advantage of contextual cues to make better predictions on objects and their relationships. The experiments show that our model significantly outperforms previous methods for generating scene graphs using Visual Genome dataset and inferring support relations with NYU Depth v2 dataset.

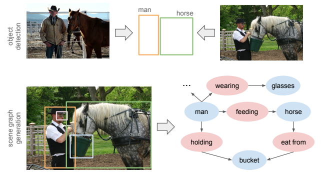


Figure 1. Object detectors perceive a scene by attending to individual objects. As a result, even a perfect detector would produce similar outputs on two semantically distinct images (first row). We propose a scene graph generation model that takes an image as input, and generates a visually-grounded scene graph (second row, right) that captures the objects in the image (blue nodes) and their pairwise relationships (red nodes).

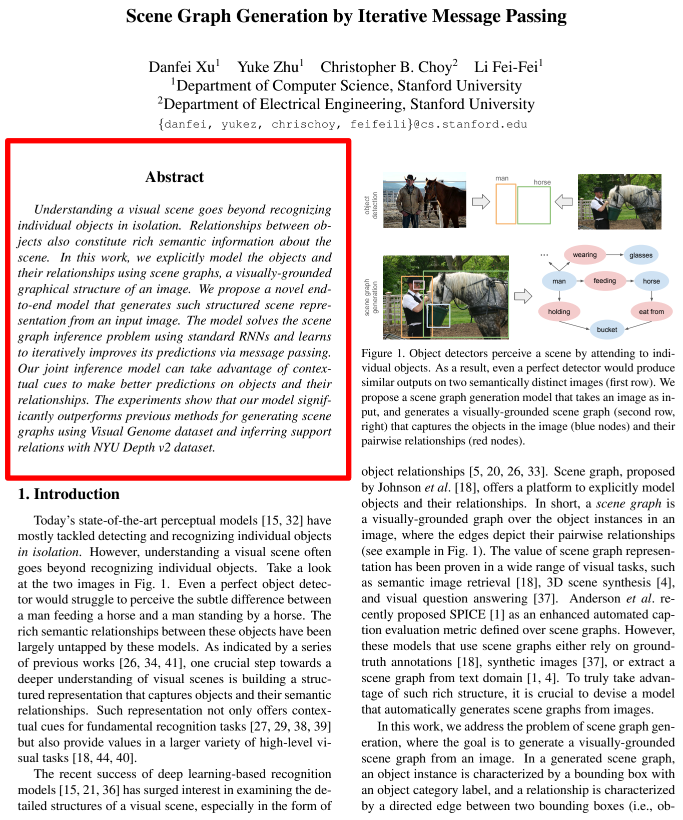
1. Introduction

Today's state-of-the-art perceptual models [15, 32] have mostly tackled detecting and recognizing individual objects *in isolation*. However, understanding a visual scene often goes beyond recognizing individual objects. Take a look at the two images in Fig. 1. Even a perfect object detector would struggle to perceive the subtle difference between a man feeding a horse and a man standing by a horse. The rich semantic relationships between these objects have been largely untapped by these models. As indicated by a series of previous works [26, 34, 41], one crucial step towards a deeper understanding of visual scenes is building a structured representation that captures objects and their semantic relationships. Such representation not only offers contextual cues for fundamental recognition tasks [27, 29, 38, 39] but also provide values in a larger variety of high-level visual tasks [18, 44, 40].

The recent success of deep learning-based recognition models [15, 21, 36] has surged interest in examining the detailed structures of a visual scene, especially in the form of object relationships [5, 20, 26, 33]. Scene graph, proposed by Johnson *et al*. [18], offers a platform to explicitly model objects and their relationships. In short, a *scene graph* is a visually-grounded graph over the object instances in an image, where the edges depict their pairwise relationships (see example in Fig. 1). The value of scene graph representation has been proven in a wide range of visual tasks, such as semantic image retrieval [18], 3D scene synthesis [4], and visual question answering [37]. Anderson *et al*. recently proposed SPICE [1] as an enhanced automated caption evaluation metric defined over scene graphs. However, these models that use scene graphs either rely on groundtruth annotations [18], synthetic images [37], or extract a scene graph from text domain [1, 4]. To truly take advantage of such rich structure, it is crucial to devise a model that automatically generates scene graphs from images.

In this work, we address the problem of scene graph generation, where the goal is to generate a visually-grounded scene graph from an image. In a generated scene graph, an object instance is characterized by a bounding box with an object category label, and a relationship is characterized by a directed edge between two bounding boxes (i.e., ob-

An anatomy of a (computer vision) paper



Scene Graph Generation by Iterative Message Passing

Danfei Xu[1] Yuke Zhu[1] Christopher B. Choy[2] Li Fei-Fei[1]
[1]Department of Computer Science, Stanford University
[2]Department of Electrical Engineering, Stanford University
{danfei, yukez, chrischoy, feifeili}@cs.stanford.edu

Abstract

Understanding a visual scene goes beyond recognizing individual objects in isolation. Relationships between objects also constitute rich semantic information about the scene. In this work, we explicitly model the objects and their relationships using scene graphs, a visually-grounded graphical structure of an image. We propose a novel end-to-end model that generates such structured scene representation from an input image. The model solves the scene graph inference problem using standard RNNs and learns to iteratively improves its predictions via message passing. Our joint inference model can take advantage of contextual cues to make better predictions on objects and their relationships. The experiments show that our model significantly outperforms previous methods for generating scene graphs using Visual Genome dataset and inferring support relations with NYU Depth v2 dataset.

1. Introduction

Today's state-of-the-art perceptual models [15, 32] have mostly tackled detecting and recognizing individual objects *in isolation*. However, understanding a visual scene often goes beyond recognizing individual objects. Take a look at the two images in Fig. 1. Even a perfect object detector would struggle to perceive the subtle difference between a man feeding a horse and a man standing by a horse. The rich semantic relationships between these objects have been largely untapped by these models. As indicated by a series of previous works [26, 34, 41], one crucial step towards a deeper understanding of visual scenes is building a structured representation that captures objects and their semantic relationships. Such representation not only offers contextual cues for fundamental recognition tasks [27, 29, 38, 39] but also provide values in a larger variety of high-level visual tasks [18, 44, 40].

The recent success of deep learning-based recognition models [15, 21, 36] has surged interest in examining the detailed structures of a visual scene, especially in the form of

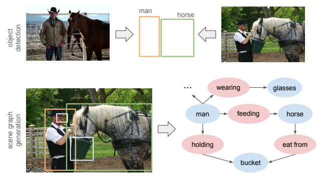


Figure 1. Object detectors perceive a scene by attending to individual objects. As a result, even a perfect detector would produce similar outputs on two semantically distinct images (first row). We propose a scene graph generation model that takes an image as input, and generates a visually-grounded scene graph (second row, right) that captures the objects in the image (blue nodes) and their pairwise relationships (red nodes).

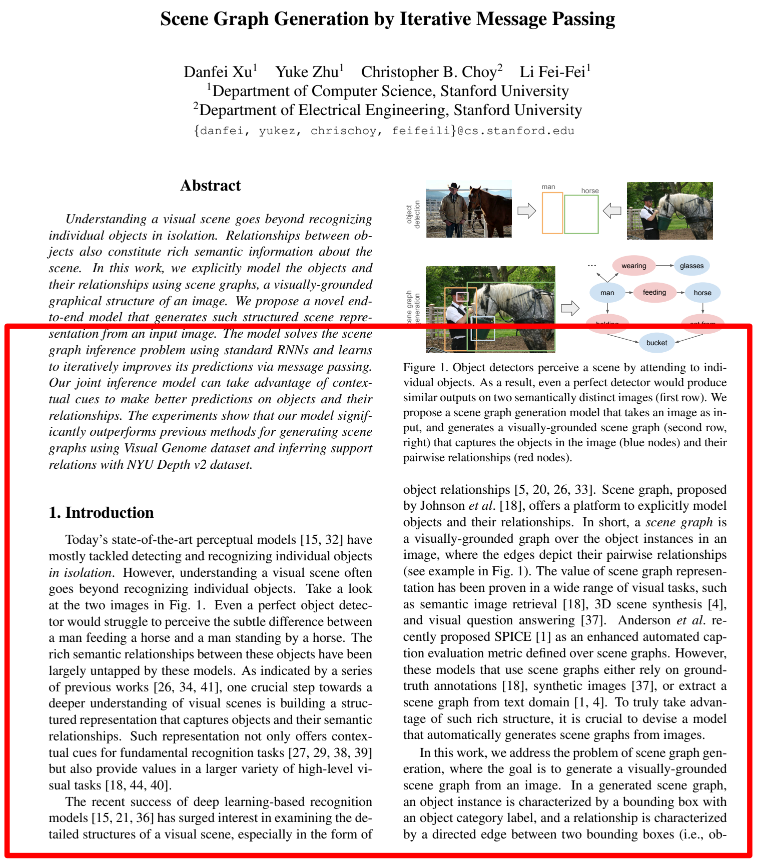
object relationships [5, 20, 26, 33]. Scene graph, proposed by Johnson *et al*. [18], offers a platform to explicitly model objects and their relationships. In short, a *scene graph* is a visually-grounded graph over the object instances in an image, where the edges depict their pairwise relationships (see example in Fig. 1). The value of scene graph representation has been proven in a wide range of visual tasks, such as semantic image retrieval [18], 3D scene synthesis [4], and visual question answering [37]. Anderson *et al*. recently proposed SPICE [1] as an enhanced automated caption evaluation metric defined over scene graphs. However, these models that use scene graphs either rely on ground-truth annotations [18], synthetic images [37], or extract a scene graph from text domain [1, 4]. To truly take advantage of such rich structure, it is crucial to devise a model that automatically generates scene graphs from images.

In this work, we address the problem of scene graph generation, where the goal is to generate a visually-grounded scene graph from an image. In a generated scene graph, an object instance is characterized by a bounding box with an object category label, and a relationship is characterized by a directed edge between two bounding boxes (i.e., ob-

Title and abstract:

- What is the paper about?
- Why should readers care about the paper?
- What's the one-sentence synopsis of the key innovation?
- What's the most exciting result?

An anatomy of a (computer vision) paper



Scene Graph Generation by Iterative Message Passing

Danfei Xu[1] Yuke Zhu[1] Christopher B. Choy[2] Li Fei-Fei[1]
[1]Department of Computer Science, Stanford University
[2]Department of Electrical Engineering, Stanford University
{danfei, yukez, chrischoy, feifeili}@cs.stanford.edu

Abstract

Understanding a visual scene goes beyond recognizing individual objects in isolation. Relationships between objects also constitute rich semantic information about the scene. In this work, we explicitly model the objects and their relationships using scene graphs, a visually-grounded graphical structure of an image. We propose a novel end-to-end model that generates such structured scene representation from an input image. The model solves the scene graph inference problem using standard RNNs and learns to iteratively improves its predictions via message passing. Our joint inference model can take advantage of contextual cues to make better predictions on objects and their relationships. The experiments show that our model significantly outperforms previous methods for generating scene graphs using Visual Genome dataset and inferring support relations with NYU Depth v2 dataset.

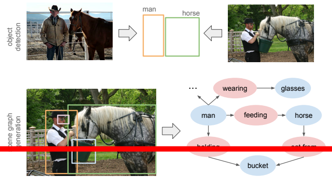


Figure 1. Object detectors perceive a scene by attending to individual objects. As a result, even a perfect detector would produce similar outputs on two semantically distinct images (first row). We propose a scene graph generation model that takes an image as input, and generates a visually-grounded scene graph (second row, right) that captures the objects in the image (blue nodes) and their pairwise relationships (red nodes).

1. Introduction

Today's state-of-the-art perceptual models [15, 32] have mostly tackled detecting and recognizing individual objects *in isolation*. However, understanding a visual scene often goes beyond recognizing individual objects. Take a look at the two images in Fig. 1. Even a perfect object detector would struggle to perceive the subtle difference between a man feeding a horse and a man standing by a horse. The rich semantic relationships between these objects have been largely untapped by these models. As indicated by a series of previous works [26, 34, 41], one crucial step towards a deeper understanding of visual scenes is building a structured representation that captures objects and their semantic relationships. Such representation not only offers contextual cues for fundamental recognition tasks [27, 29, 38, 39] but also provide values in a larger variety of high-level visual tasks [18, 44, 40].

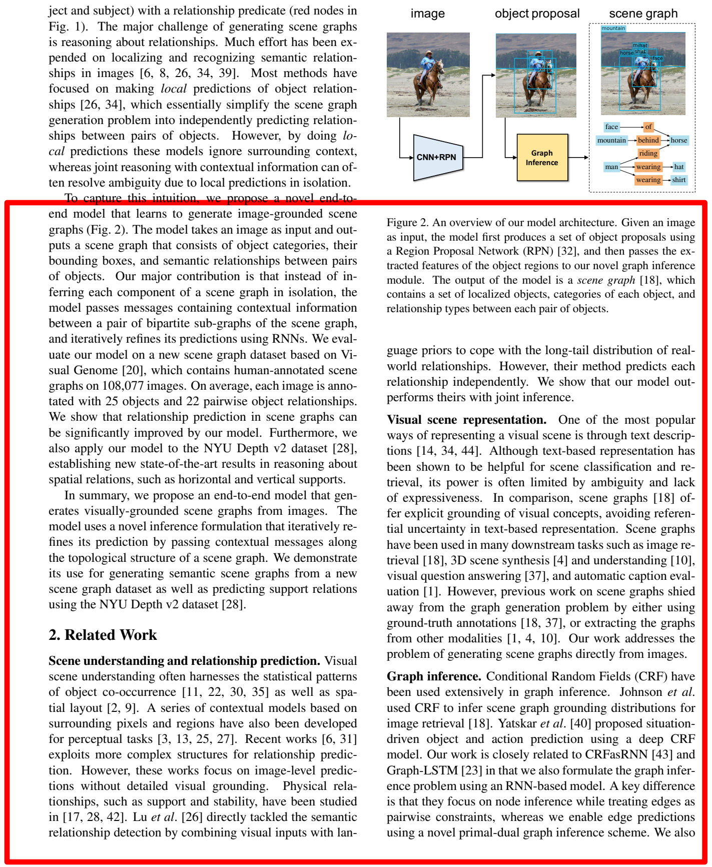
The recent success of deep learning-based recognition models [15, 21, 36] has surged interest in examining the detailed structures of a visual scene, especially in the form of object relationships [5, 20, 26, 33]. Scene graph, proposed by Johnson *et al.* [18], offers a platform to explicitly model objects and their relationships. In short, a *scene graph* is a visually-grounded graph over the object instances in an image, where the edges depict their pairwise relationships (see example in Fig. 1). The value of scene graph representation has been proven in a wide range of visual tasks, such as semantic image retrieval [18], 3D scene synthesis [4], and visual question answering [37]. Anderson *et al.* recently proposed SPICE [1] as an enhanced automated caption evaluation metric defined over scene graphs. However, these models that use scene graphs either rely on ground-truth annotations [18], synthetic images [37], or extract a scene graph from text domain [1, 4]. To truly take advantage of such rich structure, it is crucial to devise a model that automatically generates scene graphs from images.

In this work, we address the problem of scene graph generation, where the goal is to generate a visually-grounded scene graph from an image. In a generated scene graph, an object instance is characterized by a bounding box with an object category label, and a relationship is characterized by a directed edge between two bounding boxes (i.e., ob-

Introduction:

- (very high level) why is the problem important?
- What have researchers done to address the issue.
- What are their limitations?
- How is our method different?
- What is our key result?

An anatomy of a (computer vision) paper



ject and subject) with a relationship predicate (red nodes in Fig. 1). The major challenge of generating scene graphs is reasoning about relationships. Much effort has been expended on localizing and recognizing semantic relationships in images [6, 8, 26, 34, 39]. Most methods have focused on making *local* predictions of object relationships [26, 34], which essentially simplify the scene graph generation problem into independently predicting relationships between pairs of objects. However, by doing *local* predictions these models ignore surrounding context, whereas joint reasoning with contextual information can often resolve ambiguity due to local predictions in isolation.

To capture this intuition, we propose a novel end-to-end model that learns to generate image-grounded scene graphs (Fig. 2). The model takes an image as input and outputs a scene graph that consists of object categories, their bounding boxes, and semantic relationships between pairs of objects. Our major contribution is that instead of inferring each component of a scene graph in isolation, the model passes messages containing contextual information between a pair of bipartite sub-graphs of the scene graph, and iteratively refines its predictions using RNNs. We evaluate our model on a new scene graph dataset based on Visual Genome [20], which contains human-annotated scene graphs on 108,077 images. On average, each image is annotated with 25 objects and 22 pairwise object relationships. We show that relationship prediction in scene graphs can be significantly improved by our model. Furthermore, we also apply our model to the NYU Depth v2 dataset [28], establishing new state-of-the-art results in reasoning about spatial relations, such as horizontal and vertical supports.

In summary, we propose an end-to-end model that generates visually-grounded scene graphs from images. The model uses a novel inference formulation that iteratively refines its prediction by passing contextual messages along the topological structure of a scene graph. We demonstrate its use for generating semantic scene graphs from a new scene graph dataset as well as predicting support relations using the NYU Depth v2 dataset [28].

2. Related Work

Scene understanding and relationship prediction. Visual scene understanding often harnesses the statistical patterns of object co-occurrence [11, 22, 30, 35] as well as spatial layout [2, 9]. A series of contextual models based on surrounding pixels and regions have also been developed for perceptual tasks [3, 13, 25, 27]. Recent works [6, 31] exploits more complex structures for relationship prediction. However, these works focus on image-level predictions without detailed visual grounding. Physical relationships, such as support and stability, have been studied in [17, 28, 42]. Lu *et al*. [26] directly tackled the semantic relationship detection by combining visual inputs with language priors to cope with the long-tail distribution of real-world relationships. However, their method predicts each relationship independently. We show that our model outperforms theirs with joint inference.

Visual scene representation. One of the most popular ways of representing a visual scene is through text descriptions [14, 34, 44]. Although text-based representation has been shown to be helpful for scene classification and retrieval, its power is often limited by ambiguity and lack of expressiveness. In comparison, scene graphs [18] offer explicit grounding of visual concepts, avoiding referential uncertainty in text-based representation. Scene graphs have been used in many downstream tasks such as image retrieval [18], 3D scene synthesis [4] and understanding [10], visual question answering [37], and automatic caption evaluation [1]. However, previous work on scene graphs shied away from the graph generation problem by either using ground-truth annotations [18, 37], or extracting the graphs from other modalities [1, 4, 10]. Our work addresses the problem of generating scene graphs directly from images.

Graph inference. Conditional Random Fields (CRF) have been used extensively in graph inference. Johnson *et al.* used CRF to infer scene graph grounding distributions for image retrieval [18]. Yatskar *et al*. [40] proposed situation-driven object and action prediction using a deep CRF model. Our work is closely related to CRFasRNN [43] and Graph-LSTM [23] in that we also formulate the graph inference problem using an RNN-based model. A key difference is that they focus on node inference while treating edges as pairwise constraints, whereas we enable edge predictions using a novel primal-dual graph inference scheme. We also

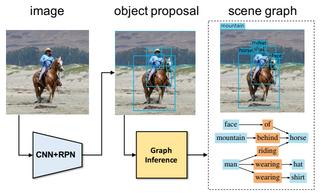
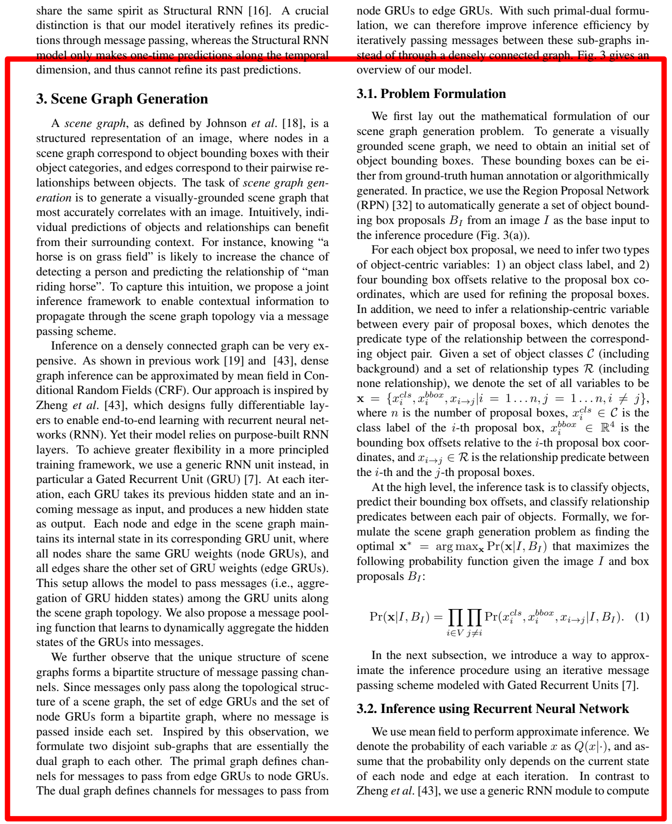


Figure 2. An overview of our model architecture. Given an image as input, the model first produces a set of object proposals using a Region Proposal Network (RPN) [32], and then passes the extracted features of the object regions to our novel graph inference module. The output of the model is a *scene graph* [18], which contains a set of localized objects, categories of each object, and relationship types between each pair of objects.

Related Work

- Who has done what?
- How is the work related to the proposed method?
- How is the proposed method different?

An anatomy of a (computer vision) paper



share the same spirit as Structural RNN [16]. A crucial distinction is that our model iteratively refines its predictions through message passing, whereas the Structural RNN model only makes one-time predictions along the temporal dimension, and thus cannot refine its past predictions.

3. Scene Graph Generation

A *scene graph*, as defined by Johnson *et al*. [18], is a structured representation of an image, where nodes in a scene graph correspond to object bounding boxes with their object categories, and edges correspond to their pairwise relationships between objects. The task of *scene graph generation* is to generate a visually-grounded scene graph that most accurately correlates with an image. Intuitively, individual predictions of objects and relationships can benefit from their surrounding context. For instance, knowing "a horse is on grass field" is likely to increase the chance of detecting a person and predicting the relationship of "man riding horse". To capture this intuition, we propose a joint inference framework to enable contextual information to propagate through the scene graph topology via a message passing scheme.

Inference on a densely connected graph can be very expensive. As shown in previous work [19] and [43], dense graph inference can be approximated by mean field in Conditional Random Fields (CRF). Our approach is inspired by Zheng *et al*. [43], which designs fully differentiable layers to enable end-to-end learning with recurrent neural networks (RNN). Yet their model relies on purpose-built RNN layers. To achieve greater flexibility in a more principled training framework, we use a generic RNN unit instead, in particular a Gated Recurrent Unit (GRU) [7]. At each iteration, each GRU takes its previous hidden state and an incoming message as input, and produces a new hidden state as output. Each node and edge in the scene graph maintains its internal state in its corresponding GRU unit, where all nodes share the same GRU weights (node GRUs), and all edges share the other set of GRU weights (edge GRUs). This setup allows the model to pass messages (i.e., aggregation of GRU hidden states) among the GRU units along the scene graph topology. We also propose a message pooling function that learns to dynamically aggregate the hidden states of the GRUs into messages.

We further observe that the unique structure of scene graphs forms a bipartite structure of message passing channels. Since messages only pass along the topological structure of a scene graph, the set of edge GRUs and the set of node GRUs form a bipartite graph, where no message is passed inside each set. Inspired by this observation, we formulate two disjoint sub-graphs that are essentially the dual graph to each other. The primal graph defines channels for messages to pass from edge GRUs to node GRUs. The dual graph defines channels for messages to pass from node GRUs to edge GRUs. With such primal-dual formulation, we can therefore improve inference efficiency by iteratively passing messages between these sub-graphs instead of through a densely connected graph. Fig. 3 gives an overview of our model.

3.1. Problem Formulation

We first lay out the mathematical formulation of our scene graph generation problem. To generate a visually grounded scene graph, we need to obtain an initial set of object bounding boxes. These bounding boxes can be either from ground-truth human annotation or algorithmically generated. In practice, we use the Region Proposal Network (RPN) [32] to automatically generate a set of object bounding box proposals B_I from an image I as the base input to the inference procedure (Fig. 3(a)).

For each object box proposal, we need to infer two types of object-centric variables: 1) an object class label, and 2) four bounding box offsets relative to the proposal box coordinates, which are used for refining the proposal boxes. In addition, we need to infer a relationship-centric variable between every pair of proposal boxes, which denotes the predicate type of the relationship between the corresponding object pair. Given a set of object classes $\mathcal{C}$ (including background) and a set of relationship types $\mathcal{R}$ (including none relationship), we denote the set of all variables to be $\mathbf{x} = \{x_i^{cls}, x_i^{bbox}, x_{i\to j} | i = 1\ldots n, j = 1\ldots n, i \neq j\}$, where n is the number of proposal boxes, $x_i^{cls} \in \mathcal{C}$ is the class label of the i-th proposal box, $x_i^{bbox} \in \mathbb{R}^4$ is the bounding box offsets relative to the i-th proposal box coordinates, and $x_{i\to j} \in \mathcal{R}$ is the relationship predicate between the i-th and the j-th proposal boxes.

At the high level, the inference task is to classify objects, predict their bounding box offsets, and classify relationship predicates between each pair of objects. Formally, we formulate the scene graph generation problem as finding the optimal $\mathbf{x}^* = \arg\max_{\mathbf{x}} \Pr(\mathbf{x}|I, B_I)$ that maximizes the following probability function given the image I and box proposals B_I:

$$\Pr(\mathbf{x}|I, B_I) = \prod_{i\in V}\prod_{j\neq i} \Pr(x_i^{cls}, x_i^{bbox}, x_{i\to j}|I, B_I). \quad (1)$$

In the next subsection, we introduce a way to approximate the inference procedure using an iterative message passing scheme modeled with Gated Recurrent Units [7].

3.2. Inference using Recurrent Neural Network

We use mean field to perform approximate inference. We denote the probability of each variable x as $Q(x|\cdot)$, and assume that the probability only depends on the current state of each node and edge at each iteration. In contrast to Zheng *et al*. [43], we use a generic RNN module to compute

Method: overview and problem setup

- What's the high-level idea of the method? What components does the method include?
- Introduce the formal problem setup
- Introduce notations

An anatomy of a (computer vision) paper

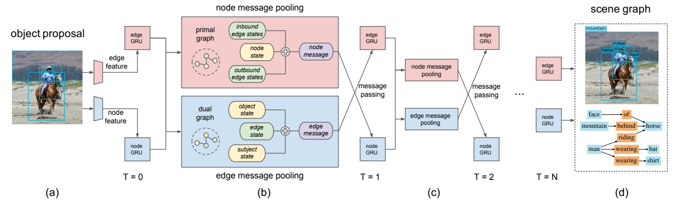


Figure 3. An illustration of our model architecture (Sec. 3). The model first extracts visual features of nodes and edges from a set of object proposals, and edge GRUs and node GRUs then take the visual features as initial input and produce a set of hidden states (a). Then a node message pooling function computes messages that are passed to the node GRU in the next iteration from the hidden states. Similarly, an edge message pooling function computes messages and feed to the edge GRU (b). The $\oplus$ symbol denotes a learnt weighted sum. The model iteratively updates the hidden states of the GRUs (c). At the last iteration step, the hidden states of the GRUs are used to predict object categories, bounding box offsets, and relationship types (d).

the hidden states. In particular, we choose Gated Recurrent Units [7] due to its simplicity and effectiveness. We use the hidden state of the corresponding GRU, a high-dimensional vector, to represent the current state of each node and each edge. As all the nodes (edges) share the same update rule, we share the same set of parameters among all the node GRUs, and the other set of parameters among all the edge GRUs (Fig. 3). We denote the current hidden state of node i as h_i and the current hidden state of edge $i \to j$ as $h_{i \to j}$. Then the mean field distribution can be formulated as

$$Q(\mathbf{x}|I, B_I) = \prod_{i=1}^{n} Q(x_i^{cls}, x_i^{bbox}|h_i)Q(h_i|f_i^v) \prod_{j \neq i} Q(x_{i \to j}|h_{i \to j})Q(h_{i \to j}|f_{i \to j}^e) \quad (2)$$

where f_i^v is the visual feature of the i-th node, and $f_{i \to j}^e$ is the visual feature of the edge from the i-th node to the j-th node. In the first iteration, the GRU units take the visual features f^v and f^e as input (Fig. 3(a)). We use the visual feature of the proposal box as the visual feature f_i^v for the i-th node. We use the visual feature of the union box over the proposal boxes b_i, b_j as the visual feature $f_{i \to j}^e$ for edge $i \in j$. These visual features are extracted by an ROI-pooling layer [12] from the image. In later iterations, the inputs are the aggregated messages from other GRU units of the previous step. We talk about how the messages are aggregated and passed in the next subsection.

3.3. Primal Dual Update and Message Pooling

Sec. 3.2 offers a generic formulation for solving graph inference problem using RNNs. However, we observe that we can further improve the inference efficiency by leveraging the unique bipartite structure of a scene graph. In the scene graph topology, the neighbors of the edge GRUs are node GRUs, and vice versa. Passing messages along this structure forms two disjoint sub-graphs that are the dual graph to each other. Specifically, we have a node-centric primal graph, in which each node GRU gets messages from its inbound and outbound edge GRUs. In the edge-centric dual graph, each edge GRU gets messages from its subject node GRU and object node GRU (Fig. 3(b)). We can therefore improve inference efficiency by iteratively passing messages between these two sub-graphs instead of through a densely connected graph (Fig. 3(c)).

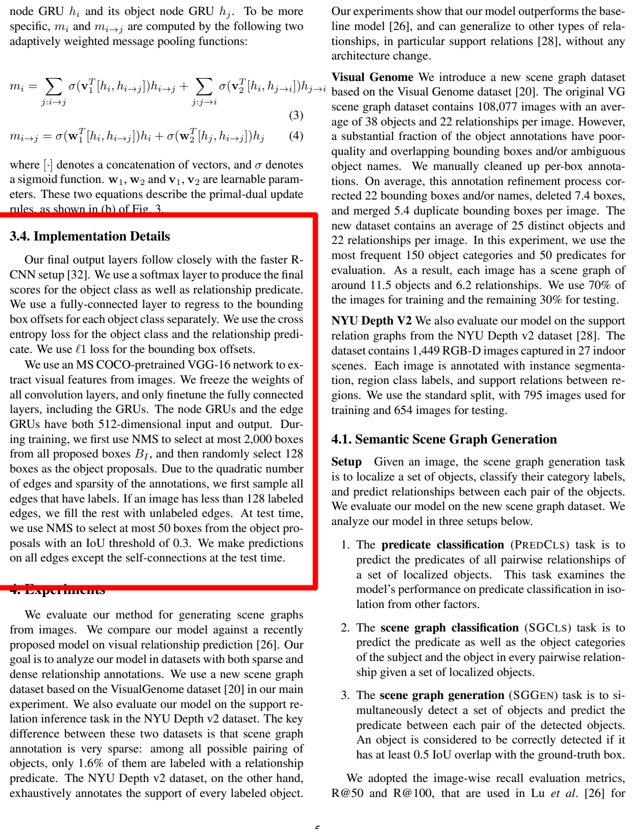
As each GRU receives multiple incoming messages, we need an aggregation function that can fuse information from all messages into a meaningful representation. A naïve approach would be standard pooling methods such as average- or max-pooling. However, we found that it is more effective to learn adaptive weights that can modulate the influences of incoming messages and only keep the relevant information. We introduce a *message pooling* function that computes the weight factors for each incoming message and fuse the messages using a weighted sum. We provide an empirical analysis of different message pooling functions in Sec. 4.

Formally, given the current GRU hidden states of nodes and edges h_i and $h_{i \to j}$, we denote the messages to update the i-th node as m_i, which is computed by a function of its own hidden state h_i, and the hidden states of its outbound edge GRUs $h_{i \to j}$ and inbound edge GRUs $h_{j \to i}$. Similarly, we denote the message to update the edge from the i-th node to the j-th node as $m_{i \to j}$, which is computed by a function of its own hidden state $h_{i \to j}$, the hidden states of its subject

Method: "the meat"

- Details of the method
- What makes the method works better than other methods, in theory.
- (Optional) new theorem, lemma
- (Optional) a figure illustration of the method

An anatomy of a (computer vision) paper



node GRU h_i and its object node GRU h_j. To be more specific, m_i and $m_{i\to j}$ are computed by the following two adaptively weighted message pooling functions:

$$m_i = \sum_{j:i\to j} \sigma(\mathbf{v}_1^T[h_i, h_{i\to j}])h_{i\to j} + \sum_{j:j\to i} \sigma(\mathbf{v}_2^T[h_i, h_{j\to i}])h_{j\to i} \tag{3}$$

$$m_{i\to j} = \sigma(\mathbf{w}_1^T[h_i, h_{i\to j}])h_i + \sigma(\mathbf{w}_2^T[h_j, h_{i\to j}])h_j \tag{4}$$

where $[\cdot]$ denotes a concatenation of vectors, and σ denotes a sigmoid function. $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{v}_1$, $\mathbf{v}_2$ are learnable parameters. These two equations describe the primal-dual update rules, as shown in (b) of Fig. 3.

3.4. Implementation Details

Our final output layers follow closely with the faster R-CNN setup [32]. We use a softmax layer to produce the final scores for the object class as well as relationship predicate. We use a fully-connected layer to regress to the bounding box offsets for each object class separately. We use the cross entropy loss for the object class and the relationship predicate. We use $\ell 1$ loss for the bounding box offsets.

We use an MS COCO-pretrained VGG-16 network to extract visual features from images. We freeze the weights of all convolution layers, and only finetune the fully connected layers, including the GRUs. The node GRUs and the edge GRUs have both 512-dimensional input and output. During training, we first use NMS to select at most 2,000 boxes from all proposed boxes B_I, and then randomly select 128 boxes as the object proposals. Due to the quadratic number of edges and sparsity of the annotations, we first sample all edges that have labels. If an image has less than 128 labeled edges, we fill the rest with unlabeled edges. At test time, we use NMS to select at most 50 boxes from the object proposals with an IoU threshold of 0.3. We make predictions on all edges except the self-connections at the test time.

4. Experiments

We evaluate our method for generating scene graphs from images. We compare our model against a recently proposed model on visual relationship prediction [26]. Our goal is to analyze our model in datasets with both sparse and dense relationship annotations. We use a new scene graph dataset based on the VisualGenome dataset [20] in our main experiment. We also evaluate our model on the support relation inference task in the NYU Depth v2 dataset. The key difference between these two datasets is that scene graph annotation is very sparse: among all possible pairing of objects, only 1.6% of them are labeled with a relationship predicate. The NYU Depth v2 dataset, on the other hand, exhaustively annotates the support of every labeled object. Our experiments show that our model outperforms the baseline model [26], and can generalize to other types of relationships, in particular support relations [28], without any architecture change.

Visual Genome We introduce a new scene graph dataset based on the Visual Genome dataset [20]. The original VG scene graph dataset contains 108,077 images with an average of 38 objects and 22 relationships per image. However, a substantial fraction of the object annotations have poor-quality and overlapping bounding boxes and/or ambiguous object names. We manually cleaned up per-box annotations. On average, this annotation refinement process corrected 22 bounding boxes and/or names, deleted 7.4 boxes, and merged 5.4 duplicate bounding boxes per image. The new dataset contains an average of 25 distinct objects and 22 relationships per image. In this experiment, we use the most frequent 150 object categories and 50 predicates for evaluation. As a result, each image has a scene graph of around 11.5 objects and 6.2 relationships. We use 70% of the images for training and the remaining 30% for testing.

NYU Depth V2 We also evaluate our model on the support relation graphs from the NYU Depth v2 dataset [28]. The dataset contains 1,449 RGB-D images captured in 27 indoor scenes. Each image is annotated with instance segmentation, region class labels, and support relations between regions. We use the standard split, with 795 images used for training and 654 images for testing.

4.1. Semantic Scene Graph Generation

Setup Given an image, the scene graph generation task is to localize a set of objects, classify their category labels, and predict relationships between each pair of the objects. We evaluate our model on the new scene graph dataset. We analyze our model in three setups below.

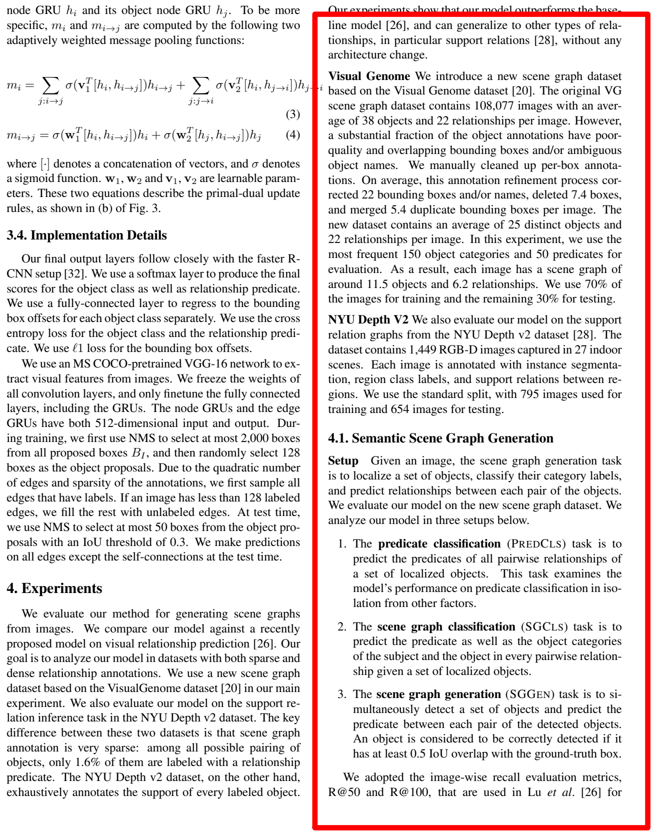
1. The **predicate classification** (PREDCLS) task is to predict the predicates of all pairwise relationships of a set of localized objects. This task examines the model's performance on predicate classification in isolation from other factors.
2. The **scene graph classification** (SGCLS) task is to predict the predicate as well as the object categories of the subject and the object in every pairwise relationship given a set of localized objects.
3. The **scene graph generation** (SGGEN) task is to simultaneously detect a set of objects and predict the predicate between each pair of the detected objects. An object is considered to be correctly detected if it has at least 0.5 IoU overlap with the ground-truth box.

We adopted the image-wise recall evaluation metrics, R@50 and R@100, that are used in Lu *et al.* [26] for

Method: implementation details

- Some details that help others to reproduce the method
- This may include (for deep learning paper): network architecture, choice of optimizers, hyperparameters, training procedure, hardware requirement.

An anatomy of a (computer vision) paper



node GRU h_i and its object node GRU h_j. To be more specific, m_i and $m_{i\to j}$ are computed by the following two adaptively weighted message pooling functions:

$$m_i = \sum_{j:i\to j} \sigma(\mathbf{v}_1^T[h_i, h_{i\to j}])h_{i\to j} + \sum_{j:j\to i} \sigma(\mathbf{v}_2^T[h_i, h_{j\to i}])h_{j\to i} \tag{3}$$

$$m_{i\to j} = \sigma(\mathbf{w}_1^T[h_i, h_{i\to j}])h_i + \sigma(\mathbf{w}_2^T[h_j, h_{i\to j}])h_j \tag{4}$$

where $[\cdot]$ denotes a concatenation of vectors, and σ denotes a sigmoid function. $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{v}_1$, $\mathbf{v}_2$ are learnable parameters. These two equations describe the primal-dual update rules, as shown in (b) of Fig. 3.

3.4. Implementation Details

Our final output layers follow closely with the faster R-CNN setup [32]. We use a softmax layer to produce the final scores for the object class as well as relationship predicate. We use a fully-connected layer to regress to the bounding box offsets for each object class separately. We use the cross entropy loss for the object class and the relationship predicate. We use $\ell 1$ loss for the bounding box offsets.

We use an MS COCO-pretrained VGG-16 network to extract visual features from images. We freeze the weights of all convolution layers, and only finetune the fully connected layers, including the GRUs. The node GRUs and the edge GRUs have both 512-dimensional input and output. During training, we first use NMS to select at most 2,000 boxes from all proposed boxes B_I, and then randomly select 128 boxes as the object proposals. Due to the quadratic number of edges and sparsity of the annotations, we first sample all edges that have labels. If an image has less than 128 labeled edges, we fill the rest with unlabeled edges. At test time, we use NMS to select at most 50 boxes from the object proposals with an IoU threshold of 0.3. We make predictions on all edges except the self-connections at the test time.

4. Experiments

We evaluate our method for generating scene graphs from images. We compare our model against a recently proposed model on visual relationship prediction [26]. Our goal is to analyze our model in datasets with both sparse and dense relationship annotations. We use a new scene graph dataset based on the VisualGenome dataset [20] in our main experiment. We also evaluate our model on the support relation inference task in the NYU Depth v2 dataset. The key difference between these two datasets is that scene graph annotation is very sparse: among all possible pairing of objects, only 1.6% of them are labeled with a relationship predicate. The NYU Depth v2 dataset, on the other hand, exhaustively annotates the support of every labeled object. Our experiments show that our model outperforms the baseline model [26], and can generalize to other types of relationships, in particular support relations [28], without any architecture change.

Visual Genome We introduce a new scene graph dataset based on the Visual Genome dataset [20]. The original VG scene graph dataset contains 108,077 images with an average of 38 objects and 22 relationships per image. However, a substantial fraction of the object annotations have poor-quality and overlapping bounding boxes and/or ambiguous object names. We manually cleaned up per-box annotations. On average, this annotation refinement process corrected 22 bounding boxes and/or names, deleted 7.4 boxes, and merged 5.4 duplicate bounding boxes per image. The new dataset contains an average of 25 distinct objects and 22 relationships per image. In this experiment, we use the most frequent 150 object categories and 50 predicates for evaluation. As a result, each image has a scene graph of around 11.5 objects and 6.2 relationships. We use 70% of the images for training and the remaining 30% for testing.

NYU Depth V2 We also evaluate our model on the support relation graphs from the NYU Depth v2 dataset [28]. The dataset contains 1,449 RGB-D images captured in 27 indoor scenes. Each image is annotated with instance segmentation, region class labels, and support relations between regions. We use the standard split, with 795 images used for training and 654 images for testing.

4.1. Semantic Scene Graph Generation

Setup Given an image, the scene graph generation task is to localize a set of objects, classify their category labels, and predict relationships between each pair of the objects. We evaluate our model on the new scene graph dataset. We analyze our model in three setups below.

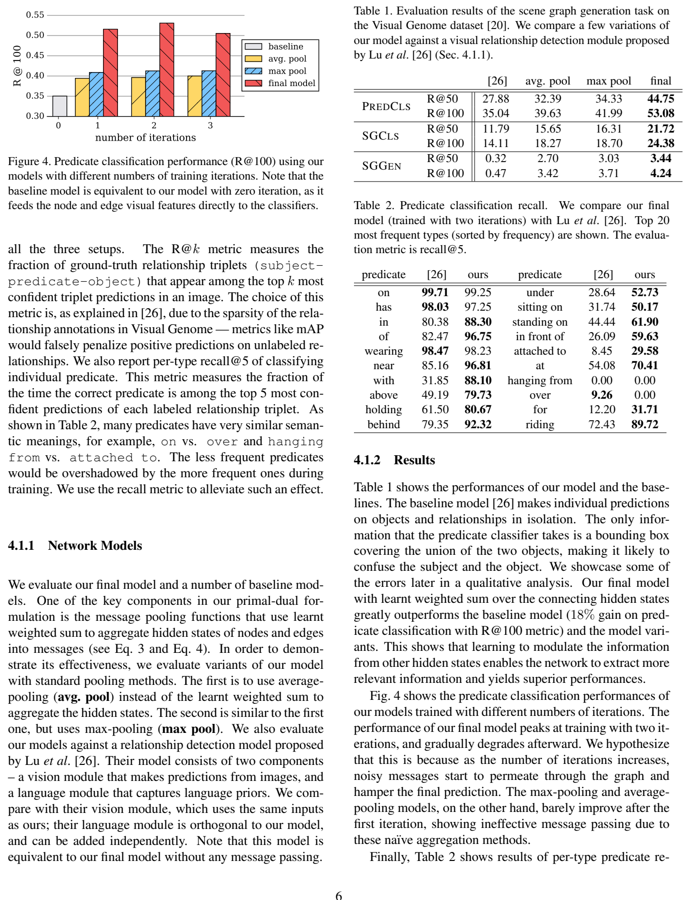
1. The **predicate classification** (PREDCLS) task is to predict the predicates of all pairwise relationships of a set of localized objects. This task examines the model's performance on predicate classification in isolation from other factors.
2. The **scene graph classification** (SGCLS) task is to predict the predicate as well as the object categories of the subject and the object in every pairwise relationship given a set of localized objects.
3. The **scene graph generation** (SGGEN) task is to simultaneously detect a set of objects and predict the predicate between each pair of the detected objects. An object is considered to be correctly detected if it has at least 0.5 IoU overlap with the ground-truth box.

We adopted the image-wise recall evaluation metrics, R@50 and R@100, that are used in Lu *et al.* [26] for

Experiment: setup

- What hypothesis are we trying to verify through the experiment?
- What dataset / environment / benchmark are we using?
- What evaluation metric are we reporting the results with?

An anatomy of a (computer vision) paper



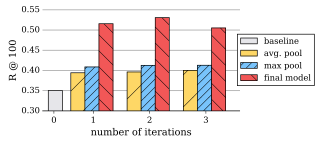


Figure 4. Predicate classification performance (R@100) using our models with different numbers of training iterations. Note that the baseline model is equivalent to our model with zero iteration, as it feeds the node and edge visual features directly to the classifiers.

all the three setups. The R@k metric measures the fraction of ground-truth relationship triplets (`subject-predicate-object`) that appear among the top k most confident triplet predictions in an image. The choice of this metric is, as explained in [26], due to the sparsity of the relationship annotations in Visual Genome — metrics like mAP would falsely penalize positive predictions on unlabeled relationships. We also report per-type recall@5 of classifying individual predicate. This metric measures the fraction of the time the correct predicate is among the top 5 most confident predictions of each labeled relationship triplet. As shown in Table 2, many predicates have very similar semantic meanings, for example, `on` vs. `over` and `hanging from` vs. `attached to`. The less frequent predicates would be overshadowed by the more frequent ones during training. We use the recall metric to alleviate such an effect.

4.1.1 Network Models

We evaluate our final model and a number of baseline models. One of the key components in our primal-dual formulation is the message pooling functions that use learnt weighted sum to aggregate hidden states of nodes and edges into messages (see Eq. 3 and Eq. 4). In order to demonstrate its effectiveness, we evaluate variants of our model with standard pooling methods. The first is to use average-pooling (**avg. pool**) instead of the learnt weighted sum to aggregate the hidden states. The second is similar to the first one, but uses max-pooling (**max pool**). We also evaluate our models against a relationship detection model proposed by Lu *et al*. [26]. Their model consists of two components – a vision module that makes predictions from images, and a language module that captures language priors. We compare with their vision module, which uses the same inputs as ours; their language module is orthogonal to our model, and can be added independently. Note that this model is equivalent to our final model without any message passing.

Table 1. Evaluation results of the scene graph generation task on the Visual Genome dataset [20]. We compare a few variations of our model against a visual relationship detection module proposed by Lu *et al*. [26] (Sec. 4.1.1).

		[26]	avg. pool	max pool	final
PREDCLS	R@50	27.88	32.39	34.33	**44.75**
	R@100	35.04	39.63	41.99	**53.08**
SGCLS	R@50	11.79	15.65	16.31	**21.72**
	R@100	14.11	18.27	18.70	**24.38**
SGGEN	R@50	0.32	2.70	3.03	**3.44**
	R@100	0.47	3.42	3.71	**4.24**

Table 2. Predicate classification recall. We compare our final model (trained with two iterations) with Lu *et al*. [26]. Top 20 most frequent types (sorted by frequency) are shown. The evaluation metric is recall@5.

predicate	[26]	ours	predicate	[26]	ours
on	**99.71**	99.25	under	28.64	**52.73**
has	**98.03**	97.25	sitting on	31.74	**50.17**
in	80.38	**88.30**	standing on	44.44	**61.90**
of	82.47	**96.75**	in front of	26.09	**59.63**
wearing	**98.47**	98.23	attached to	8.45	**29.58**
near	85.16	**96.81**	at	54.08	**70.41**
with	31.85	**88.10**	hanging from	0.00	0.00
above	49.19	**79.73**	over	**9.26**	0.00
holding	61.50	**80.67**	for	12.20	**31.71**
behind	79.35	**92.32**	riding	72.43	**89.72**

4.1.2 Results

Table 1 shows the performances of our model and the baselines. The baseline model [26] makes individual predictions on objects and relationships in isolation. The only information that the predicate classifier takes is a bounding box covering the union of the two objects, making it likely to confuse the subject and the object. We showcase some of the errors later in a qualitative analysis. Our final model with learnt weighted sum over the connecting hidden states greatly outperforms the baseline model (18% gain on predicate classification with R@100 metric) and the model variants. This shows that learning to modulate the information from other hidden states enables the network to extract more relevant information and yields superior performances.

Fig. 4 shows the predicate classification performances of our models trained with different numbers of iterations. The performance of our final model peaks at training with two iterations, and gradually degrades afterward. We hypothesize that this is because as the number of iterations increases, noisy messages start to permeate through the graph and hamper the final prediction. The max-pooling and average-pooling models, on the other hand, barely improve after the first iteration, showing ineffective message passing due to these naïve aggregation methods.

Finally, Table 2 shows results of per-type predicate re-

Experiment: results

- **Quantitative** results: tables, barcharts, plots, …
- Discuss what the results mean to our hypothesis.

An anatomy of a (computer vision) paper

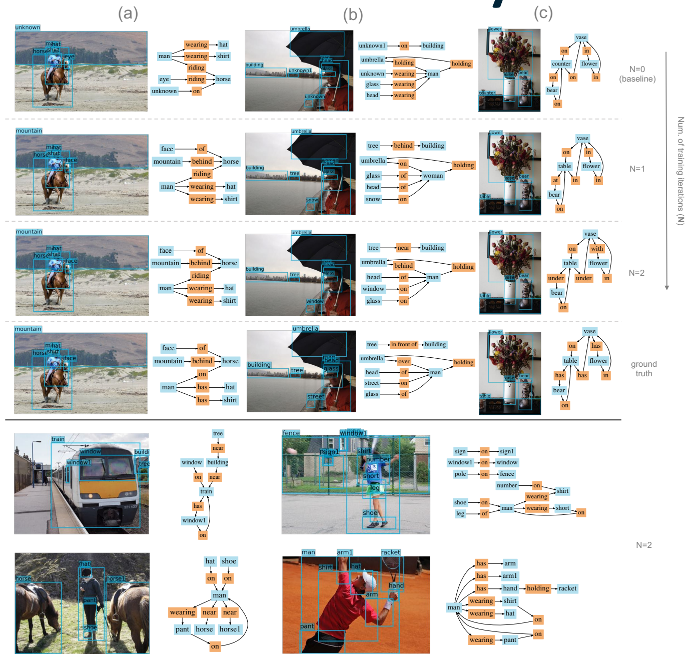


Figure 5. Sample predictions from the baseline model and our final model trained with different numbers of message passing iterations. The models take images and object bounding boxes as input, and produce object class labels (blue boxes) and relationship predicates between each pair of objects (orange boxes). In order to keep the visualization interpretable, we only show the relationship (edge) predictions for the pairs of objects (nodes) that have ground-truth relationship annotations.

call. Both the baseline model and our final model perform well in predicting frequent predicates. However, the gap between the models expands for less frequent predicates. This is because our model uses contextual information to cope with the uneven distribution in the relationship annotations, whereas the baseline model suffers more from the skewed distribution by making predictions in isolation.

4.1.3 Qualitative results

Fig. 5 shows qualitative results that compare our final model trained with different numbers of iterations and the baseline model. The results show that the baseline model tends to confuse about the subject and the object in a relationship. For example, it predicts `(umbrella-holding-man)` in (b) and `(counter-on-vase)` in (c). Our fi-

Experiment: results

- **Qualitative** results: show examples of the system input / output
- Discuss what the results mean to our hypothesis.

An anatomy of a (computer vision) paper

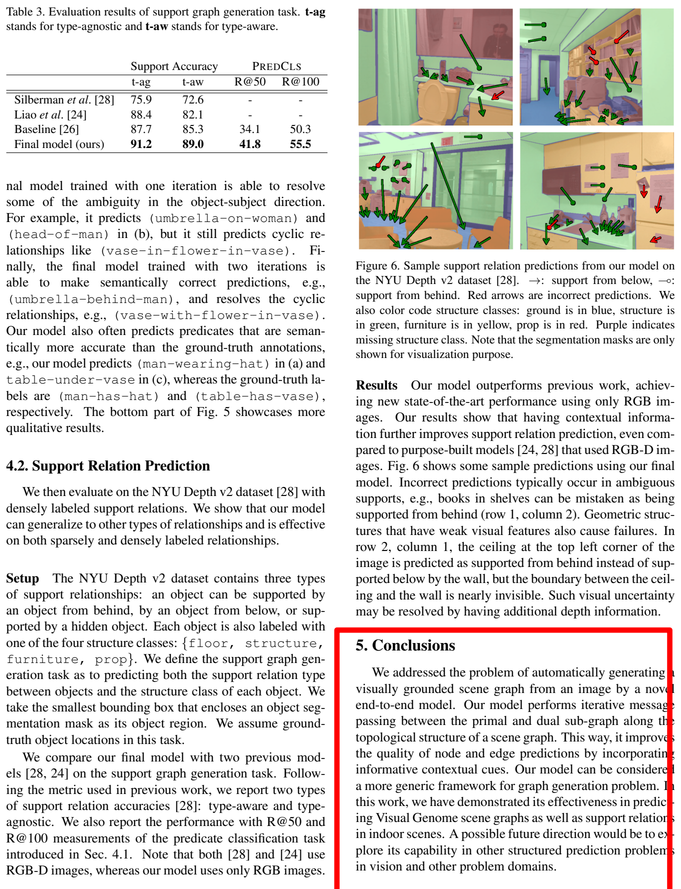


Table 3. Evaluation results of support graph generation task. **t-ag** stands for type-agnostic and **t-aw** stands for type-aware.

	Support Accuracy		PredCls	
	t-ag	t-aw	R@50	R@100
Silberman *et al.* [28]	75.9	72.6	-	-
Liao *et al.* [24]	88.4	82.1	-	-
Baseline [26]	87.7	85.3	34.1	50.3
Final model (ours)	**91.2**	**89.0**	**41.8**	**55.5**

nal model trained with one iteration is able to resolve some of the ambiguity in the object-subject direction. For example, it predicts (umbrella-on-woman) and (head-of-man) in (b), but it still predicts cyclic relationships like (vase-in-flower-in-vase). Finally, the final model trained with two iterations is able to make semantically correct predictions, e.g., (umbrella-behind-man), and resolves the cyclic relationships, e.g., (vase-with-flower-in-vase). Our model also often predicts predicates that are semantically more accurate than the ground-truth annotations, e.g., our model predicts (man-wearing-hat) in (a) and table-under-vase in (c), whereas the ground-truth labels are (man-has-hat) and (table-has-vase), respectively. The bottom part of Fig. 5 showcases more qualitative results.

4.2. Support Relation Prediction

We then evaluate on the NYU Depth v2 dataset [28] with densely labeled support relations. We show that our model can generalize to other types of relationships and is effective on both sparsely and densely labeled relationships.

Setup The NYU Depth v2 dataset contains three types of support relationships: an object can be supported by an object from behind, by an object from below, or supported by a hidden object. Each object is also labeled with one of the four structure classes: {floor, structure, furniture, prop}. We define the support graph generation task as to predicting both the support relation type between objects and the structure class of each object. We take the smallest bounding box that encloses an object segmentation mask as its object region. We assume ground-truth object locations in this task.

We compare our final model with two previous models [28, 24] on the support graph generation task. Following the metric used in previous work, we report two types of support relation accuracies [28]: type-aware and type-agnostic. We also report the performance with R@50 and R@100 measurements of the predicate classification task introduced in Sec. 4.1. Note that both [28] and [24] use RGB-D images, whereas our model uses only RGB images.

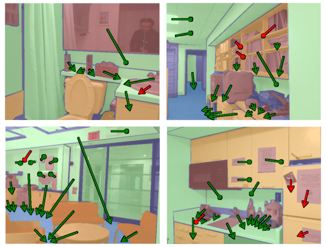


Figure 6. Sample support relation predictions from our model on the NYU Depth v2 dataset [28]. →: support from below, ⊸: support from behind. Red arrows are incorrect predictions. We also color code structure classes: ground is in blue, structure is in green, furniture is in yellow, prop is in red. Purple indicates missing structure class. Note that the segmentation masks are only shown for visualization purpose.

Results Our model outperforms previous work, achieving new state-of-the-art performance using only RGB images. Our results show that having contextual information further improves support relation prediction, even compared to purpose-built models [24, 28] that used RGB-D images. Fig. 6 shows some sample predictions using our final model. Incorrect predictions typically occur in ambiguous supports, e.g., books in shelves can be mistaken as being supported from behind (row 1, column 2). Geometric structures that have weak visual features also cause failures. In row 2, column 1, the ceiling at the top left corner of the image is predicted as supported from behind instead of supported below by the wall, but the boundary between the ceiling and the wall is nearly invisible. Such visual uncertainty may be resolved by having additional depth information.

5. Conclusions

We addressed the problem of automatically generating a visually grounded scene graph from an image by a novel end-to-end model. Our model performs iterative message passing between the primal and dual sub-graph along the topological structure of a scene graph. This way, it improves the quality of node and edge predictions by incorporating informative contextual cues. Our model can be considered a more generic framework for graph generation problem. In this work, we have demonstrated its effectiveness in predicting Visual Genome scene graphs as well as support relations in indoor scenes. A possible future direction would be to explore its capability in other structured prediction problems in vision and other problem domains.

Discussion & conclusions

- Reiterate the key message
- Main takeaways
- Limitations / future works

The first pass: a quick scan

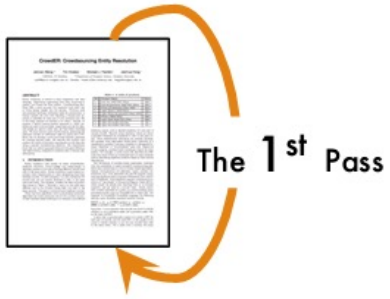
Goal: get bird's-eye view of the paper (5~10 min)

What to read:

- Title, abstract, introduction and conclusion
- Section and sub-section headings
- Main figures
- Scan of bibliography

You should be able to answer:

- What type of paper is this?
- What are the main contributions?



Let’s try it!

1. **Category**: What type of paper is this? A new algorithm paper? An analysis of existing algorithm? A benchmark paper?
2. **Context**: Which other papers is it related to? Which problem setup were used to analyze the problem?
3. **Correctness**: Do the assumptions appear to be valid?
4. **Contributions**: What are the paper’s main contributions?
5. **Clarity**: Is the paper well written?

The second pass: grasp the content

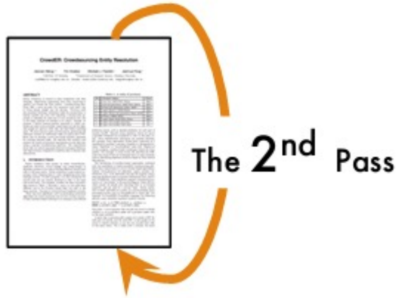
Goal: get a good understanding of the ”meat” of the paper.

How to read:

- Look carefully at figures, diagrams and examples
- Take notes of questions, unread references etc.
- Ignore proofs, appendix, extensions etc.

You should be able to:

- Summarize main thrusts of the paper, with supporting evidence, to someone else



The third pass: all about the details

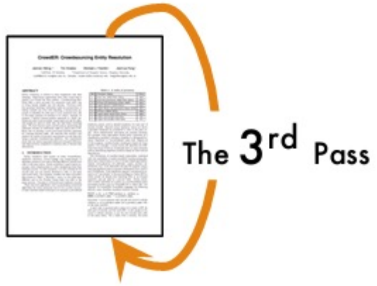
Goal: think about what you would have done if you were to re-implement such an idea

How to read:

- Challenge every assumption
- Compare your version with the actual paper
  - Often leads to questions like: why not do it this way?

You should be able to:

- Identify hidden assumptions/potential design flaws
- Get ideas for future work



How to do a literature review

1. Pick your favorite academic search engine (e.g., Google scholar) and start with keywords
2. Find 3-5 recent and highly cited papers
    - From reputable venues and by reputable institution/author
    - If you find a survey paper, start from the survey paper
3. Do the first pass to identify key papers and researchers that these works cite
4. Track down these papers/researchers
5. Iterate as needed

Tools

Backward influence: influential citations in the papers that you've read

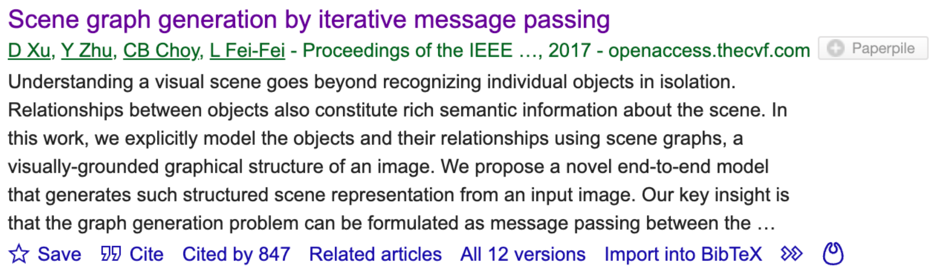
How: reading

Forward influence: papers citing the ones that you've read

How : Google Scholar's "Cited By"

Relatedness: contemporaneous but not citing

How: Google Scholar's "Related articles"



Scene graph generation by iterative message passing

D Xu, Y Zhu, CB Choy, L Fei-Fei - Proceedings of the IEEE …, 2017 - openaccess.thecvf.com

Understanding a visual scene goes beyond recognizing individual objects in isolation. Relationships between objects also constitute rich semantic information about the scene. In this work, we explicitly model the objects and their relationships using scene graphs, a visually-grounded graphical structure of an image. We propose a novel end-to-end model that generates such structured scene representation from an input image. Our key insight is that the graph generation problem can be formulated as message passing between the …

Top conferences and journals

* Just search for “<acronym> + conference”

Machine Learning

NeurIPS, ICLR, ICML, AISTATS, AAAI, JMLR (journal)

Computer Vision

CVPR, ECCV, ICCV, BMVC, T-PAMI (journal), IJCV (journal)

Natural Language Processing

ACL, EMNLP, NAACL,

Robotics:

RSS, ICRA, IROS, CORL, T-RO (journal), IJRR (journal)

NEXT TIME

Convolution

Convolutional Neural Networks