Graph Convolutional Neural Networks for Web-Scale Recommender Systems

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, Jure Leskovec

Raphaël Fournier-S'niehotta

Journal-club MSDMA

19 octobre 2018

Plan

1 Introduction

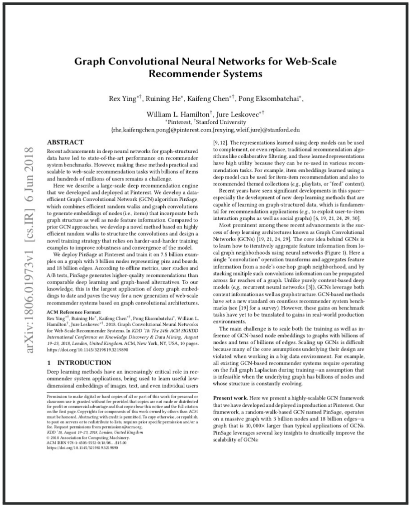
2 Graph convolutional networks

3 PinSage

4 Experiments

Introduction

- published at KDD'18
- team leader: Jure Leskovec, from Stanford (and Pinterest)
- follows their recent works on GCN (=GCNN) [**HamiltonYL17NIPS**; **HYLtutoWWW**; HYL17]



arXiv:1806.01973v1 [cs.IR] 6 Jun 2018

Graph Convolutional Neural Networks for Web-Scale Recommender Systems

Rex Ying*†, Ruining He*, Kaifeng Chen*†, Pong Eksombatchai*,

William L. Hamilton†, Jure Leskovec*†

*Pinterest, †Stanford University

{rhe,kaifengchen,pong}@pinterest.com,{rexying,wleif,jure}@stanford.edu

ABSTRACT

Recent advancements in deep neural networks for graph-structured data have led to state-of-the-art performance on recommender system benchmarks. However, making these methods practical and scalable to web-scale recommendation tasks with billions of items and hundreds of millions of users remains a challenge.

Here we describe a large-scale deep recommendation engine that we developed and deployed at Pinterest. We develop a data-efficient Graph Convolutional Network (GCN) algorithm PinSage, which combines efficient random walks and graph convolutions to generate embeddings of nodes (i.e., items) that incorporate both graph structure as well as node feature information. Compared to prior GCN approaches, we develop a novel method based on highly efficient random walks to structure the convolutions and design a novel training strategy that relies on harder-and-harder training examples to improve robustness and convergence of the model.

We deploy PinSage at Pinterest and train it on 7.5 billion examples on a graph with 3 billion nodes representing pins and boards, and 18 billion edges. According to offline metrics, user studies and A/B tests, PinSage generates higher-quality recommendations than comparable deep learning and graph-based alternatives. To our knowledge, this is the largest application of deep graph embeddings to date and paves the way for a new generation of web-scale recommender systems based on graph convolutional architectures.

ACM Reference Format:

Rex Ying*†, Ruining He*, Kaifeng Chen*†, Pong Eksombatchai*, William L. Hamilton†, Jure Leskovec*†. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219890

1 INTRODUCTION

Deep learning methods have an increasingly critical role in recommender system applications, being used to learn useful low-dimensional embeddings of images, text, and even individual users [9, 12]. The representations learned using deep models can be used to complement, or even replace, traditional recommendation algorithms like collaborative filtering. and these learned representations have high utility because they can be re-used in various recommendation tasks. For example, item embeddings learned using a deep model can be used for item-item recommendation and also to recommended themed collections (*e.g.*, playlists, or "feed" content).

Recent years have seen significant developments in this space—especially the development of new deep learning methods that are capable of learning on graph-structured data, which is fundamental for recommendation applications (*e.g.*, to exploit user-to-item interaction graphs as well as social graphs) [6, 19, 21, 24, 29, 30].

Most prominent among these recent advancements is the success of deep learning architectures known as Graph Convolutional Networks (GCNs) [19, 21, 24, 29]. The core idea behind GCNs is to learn how to iteratively aggregate feature information from local graph neighborhoods using neural networks (Figure 1). Here a single "convolution" operation transforms and aggregates feature information from a node's one-hop graph neighborhood, and by stacking multiple such convolutions information can be propagated across far reaches of a graph. Unlike purely content-based deep models (*e.g.*, recurrent neural networks [3]), GCNs leverage both content information as well as graph structure. GCN-based methods have set a new standard on countless recommender system benchmarks (see [19] for a survey). However, these gains on benchmark tasks have yet to be translated to gains in real-world production environments.

The main challenge is to scale both the training as well as inference of GCN-based node embeddings to graphs with billions of nodes and tens of billions of edges. Scaling up GCNs is difficult because many of the core assumptions underlying their design are violated when working in a big data environment. For example, all existing GCN-based recommender systems require operating on the full graph Laplacian during training—an assumption that is infeasible when the underlying graph has billions of nodes and whose structure is constantly evolving.

Present work. Here we present a highly-scalable GCN framework that we have developed and deployed in production at Pinterest. Our framework, a random-walk-based GCN named PinSage, operates on a massive graph with 3 billion nodes and 18 billion edges—a graph that is $10,000\times$ larger than typical applications of GCNs. PinSage leverages several key insights to drastically improve the scalability of GCNs:

KDD '18, August 19–23, 2018, London, United Kingdom
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219890

Main contributions

- very large scale recommender system
- deployed in production
- new Graph Convolutional NN algorithm
- efficient training strategy (locality / choice of examples)

Plan

1 Introduction

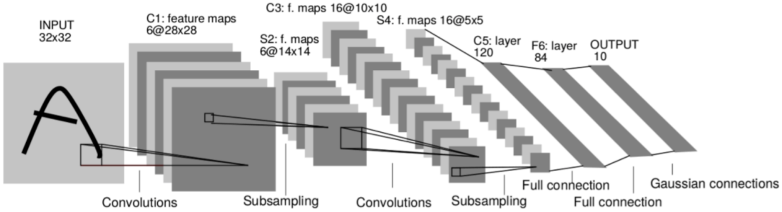
2 Graph convolutional networks

3 PinSage

4 Experiments

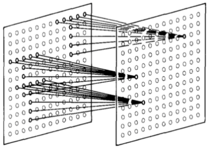
Convolutional Neural Networks

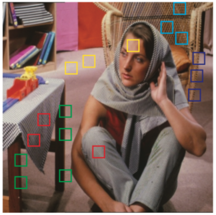
- An architecture for high-dimensional learning

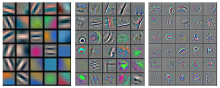


ConvNets

- Assumption: data (video, sound, image) are compositional, formed of patterns that are:
  - local (*c.f.* visual neurons)
  - stationary (global/local invariance)
  - multi-scale (hierarchy)
- ConvNets extract compositional features and feed them to classifier, recommender, etc.



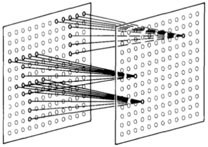


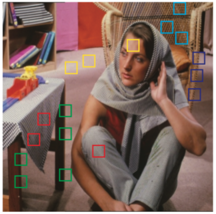


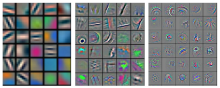
Slide material from Xavier Bresson @ IPAM

ConvNets

- Assumption: data (video, sound, image) are compositional, formed of patterns that are:
  - local (*c.f.* visual neurons) $O(1)$ parameters per filter
  - stationary (global/local invariance) $O(n \log n)$ with FFT
  - multi-scale (hierarchy) $O(n)$ downsampling & pooling
- ConvNets extract compositional features and feed them to classifier, recommender, etc.







Slide material from Xavier Bresson @ IPAM

Challenges with graphs

Graph data

- non euclidian
- limited engineered features (inflexible)

Representation learning: extend CNN to graphs

- node embeddings
- Assumption: Non euclidian data are still stationary and have hierarchy
- Define convolution and pooling for graphs (compositionality)
    - Convolution: spectral graph theory
    - Downsampling with clustering techniques
- Fast computations?

Slide material partially from Xavier Bresson @ IPAM

State of the art on GCNs

- Seminal work on neural nets for graph data: [GMS05; Sca+09]
- Creation of GCNs in [Bru+13]
- Several extensions of spectral convolutions, with applications in different domains [KW16; MBB17]
- Several extensions of spectral convolutions, with applications in different domains [KW16; MBB17]

State of the art on GCNs

- Seminal work on neural nets for graph data: [GMS05; Sca+09] Problem: message passing too expensive
- Creation of GCNs in [Bru+13]
- Several extensions of spectral convolutions, with applications in different domains [KW16; MBB17]
- Several extensions of spectral convolutions, with applications in different domains [KW16; MBB17]
  All beating matrix facto or random-walk -based approaches (node2vec, DeepWalk)
- Good survey in [HYL17; Bro+17]
- Scalability problem remains

Plan

1 Introduction

2 Graph convolutional networks

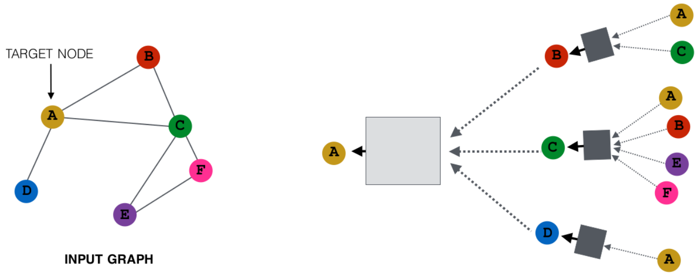
3 PinSage

4 Experiments

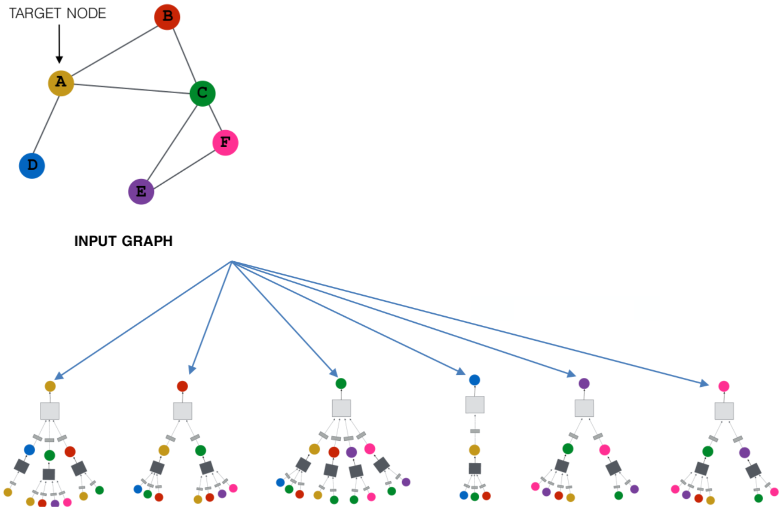
PinSage Architecture

Assume we have a graph G

- V is the vertex set
- A is the adjacency matrix
- $X \in \mathbb{R}^{m \times |V|}$ is a matrix of node features (text, image data, but also node degrees, cluster coefficients)
- idea: generate node embeddings (neighborhood info) with neural networks

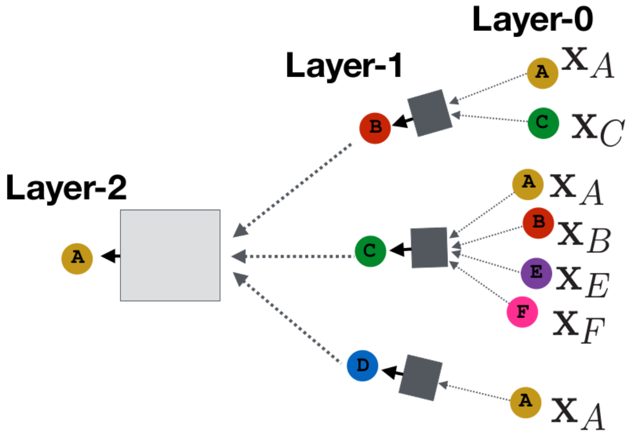


Neighborhood aggregation



Neighborhood aggregation: layers

- nodes have embeddings at each layer
- layer 0 of node v is the feature vector x_v



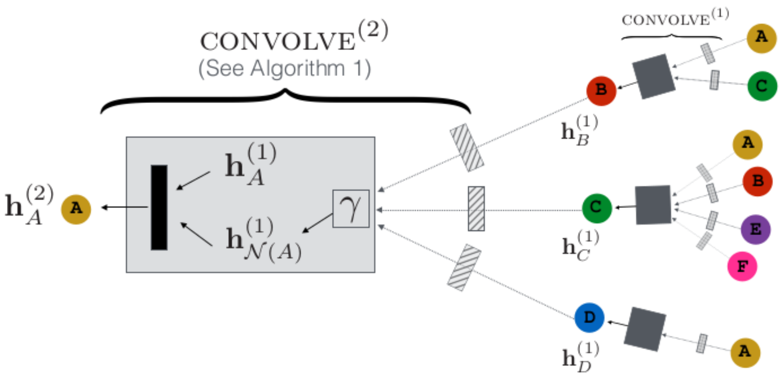
Node embeddings

$$\boldsymbol{h}_v^k = \sigma\left(\boldsymbol{W}_k \sum_{u\in N(v)} \frac{\boldsymbol{h}_u^{k-1}}{|N(v)|} + \boldsymbol{B}_k \boldsymbol{h}_v^{k-1}\right), \forall k > 0$$

- $\boldsymbol{h}_v^k$: k^{th} layer embedding for v
- σ: ReLU
- $\sum_{u\in N(v)} \frac{\boldsymbol{h}_u^k-1}{|N(v)|}$ average neighbors' previous layers embeddings
- $\boldsymbol{W}_k, \boldsymbol{B}_k$ trainable matrices (weights, bias)

- $\boldsymbol{h}_{\boldsymbol{v}}^0 = x_v$
- $z_v = h_v^K$, $K = 2$

Architecture schema



Idea

- Generalized aggregation: Replace the simple average by a different *pooling* method
- Item-wise mean/max, γ

Random walks

- Sampling neighborhoods with random walks
- top T nodes with best Personalized PageRank
- Advantages:
  - avoid storing all Laplacian in memory
  - fixed memory footprint
  - shared parameters between subgraphs
  - importance encoded in embedding aggregations

Training

- Supervised max-margin ranking loss
- Idea: maximize inner product of positive examples (proximity between embeddings of *q* and *i*)
- and: product between query and negative example must be smaller than any positive example by pre-defined margin

$$\mathscr{L} = \sum_{(q,i)\in\mathscr{D}} max(0, -z_q^T z_i + z_q^T z_{neg} + \Delta)$$

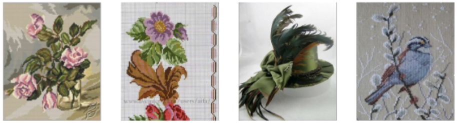
- Δ: margin

Negative sampling: a challenge

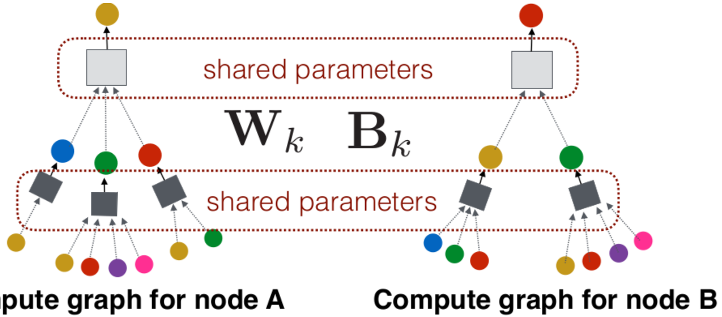
- cannot be uniform (resolution too low)
- cannot be individual (too expensive)
- batching, fixed 500 negative samples per batch

Curriculum training [Ben+09]

- needle in haystack: find 1000 similar items to query in 2B
- 500 in 2B is too low: bad parameters updates
- find **hard negative samples**
- between rank 2000-5000 in PPR with q

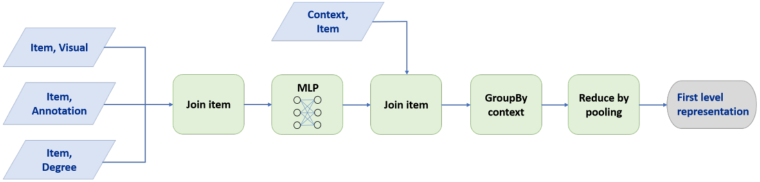


Inductive capability



Last idea: MapReduce computations

- problem: expensive computation, due to overlap



- offline embedding computation
- producer-consumer CPU/GPU framework
- recommendation by lookups in the embedding space
- LSH-based retrieval
- online recommendation served!

Plan

1 Introduction

2 Graph convolutional networks

3 PinSage

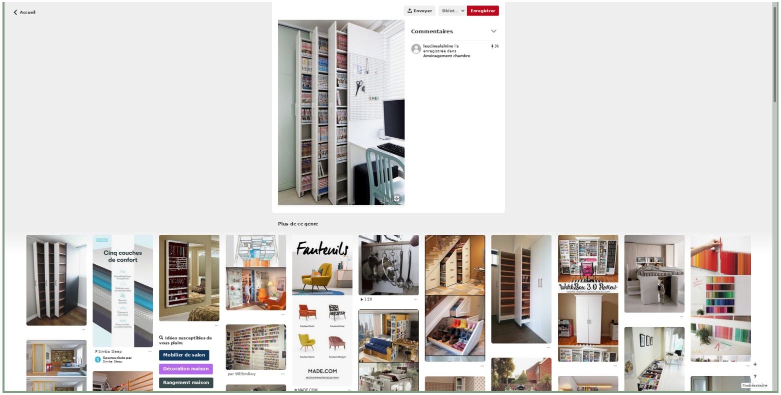
4 Experiments
- Experimental setup
- Evaluation results

Application: recommender system

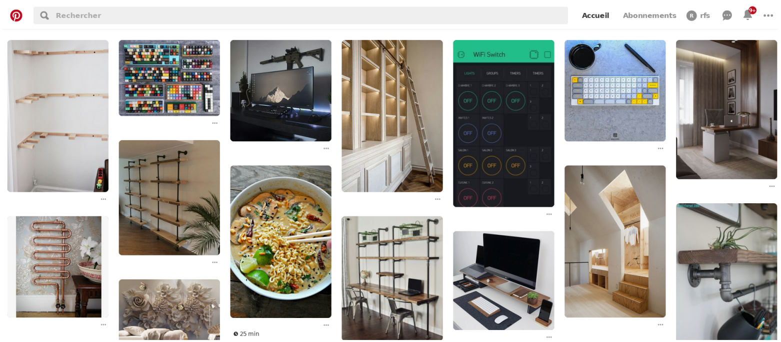
Specific tasks, not exactly regular ones

- recommending related pins (item similarity)
- recommend pins for user’s home/feed

Related pin after image query



Home feed



Dataset

- positive examples: use history of interactions to create (q, i) pairs (query image q, next pin i)
- all other pins are considered negative

- 1.2 billions positive pairs use for training
- 6 hard negative items per pin
- 500 negatives per batch
- total: 7.5 billions items

Dataset: graph sampling

- PinSage efficiently generate embeddings for unseen data
- training on 20% of all boards (and all their pins)
- 70% of labeled examples
- 10% more during hyperparameter tuning
- remaining 20% used for testing in offline evaluations

- full datatsets: 18TB
- 4TB output embeddings

Features

- Each pin has image and (title, annotation)
- visual embeddings (dim 4096) (VGG-16 architecture 6th layer [SZ14])
- textual annotations embeddings (dim 256, Word2vec)
- log degree (only one direct graph feature)

Variants

- max-pooling γ
- mean-pooling γ
- mean-pooling with cross-entropy loss (previous work)
- mean-pooling with hard negative samples

- $K = 2$
- $m = 2048$
- embedding dimension $d = 1024$

Computation resources

- TensorFlow implementation
- Training on a single machine with:
  - 16 Tesla K80 GPU
  - 32 cores
  - Linux HugePages
  - 500 GB Mem for training
- MapReduce on AWS 378 nodes Hadoop cluster

Offline Evaluation

- for each (q, i) pair in test set, compute $K = 500$ nearest neighbors of q among 5 million test pins
- *hit rate*: fraction of queries where i was ranked in the NN
- Mean Reciprocal Rank MRR:

$$\mathrm{MRR} = \frac{1}{n} \sum_{(q,i)} \frac{1}{\lceil R_{i,q}/100 \rceil}$$

(scaling w/ factor 100 insures diff at rank 1000 and at rank 2000 are significant)

- PinSage beats all variants, hit rate $\sim 67\%$, MRR 0.59 (second: mean pooling hard)
- Also: checked that embeddings similarities is sufficiently distributed, so that there is enough resolution to distinguish between items, LSH collision probabilities are low

User studies

Head to head

- image of query pin presented to user, with 2 images from different algorithms
- 2/3 consensus between users
- PinSage vs Baseline is $\sim 50\%$ draws.
- but when users have an opinion, it's for PinSage (approx. 60% of wins)

A/B tests

- metric: repin rate (home feed recos saved by users)
- 10-30% improvement over baselines

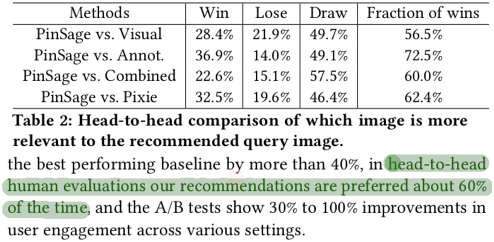
Bonus

GCN can be inductive so:

- training on subgraph (instead of full graph)
- easy to compute embeddings for new nodes (cold-start problem)

Critiques

- Limited graph features (node degree, and PPR proxy)
- Performance claims...



Methods	Win	Lose	Draw	Fraction of wins
PinSage vs. Visual	28.4%	21.9%	49.7%	56.5%
PinSage vs. Annot.	36.9%	14.0%	49.1%	72.5%
PinSage vs. Combined	22.6%	15.1%	57.5%	60.0%
PinSage vs. Pixie	32.5%	19.6%	46.4%	62.4%

Table 2: Head-to-head comparison of which image is more relevant to the recommended query image.

the best performing baseline by more than 40%, in head-to-head human evaluations our recommendations are preferred about 60% of the time, and the A/B tests show 30% to 100% improvements in user engagement across various settings.

Conclusion

- Random-walk based GCN
- Highly scalable (10 000x !)
- Performance improved by:
  - importance pooling (PageRank like sampling)
  - curriculum training (harder and harder examples)
- Reusable embeddings
- Comprehensive evaluation (possible thanks to production context)

Contact

Thank you for your attention.

Contact: fournier@cnam.fr

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. “Curriculum learning”. In: *Proceedings of the 26th annual international conference on machine learning*. ACM. 2009, pp. 41–48.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. “Geometric deep learning: going beyond euclidean data”. In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. “Spectral networks and locally connected networks on graphs”. In: *arXiv preprint arXiv:1312.6203* (2013).

Marco Gori, Gabriele Monfardini, and Franco Scarselli. “A new model for learning in graph domains”. In: *Neural Networks, 2005. IJCNN’05. Proceedings. 2005 IEEE International Joint Conference on*. Vol. 2. IEEE. 2005, pp. 729–734.

William L. Hamilton, Rex Ying, and Jure Leskovec. “Representation Learning on Graphs: Methods and Applications”. In: *IEEE Data Eng. Bull.* 40.3 (2017), pp. 52–74. URL: http://sites.computer.org/debull/A17sept/p52.pdf.

Thomas N Kipf and Max Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).

Federico Monti, Michael Bronstein, and Xavier Bresson. “Geometric matrix completion with recurrent multi-graph neural networks”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 3697–3707.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. “The graph neural network model”. In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80.

Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).