Massachusetts Institute of Technology

Transfer Learning and Neural Networks for Graphs

Stefanie Jegelka
MIT EECS

September 24, 2021

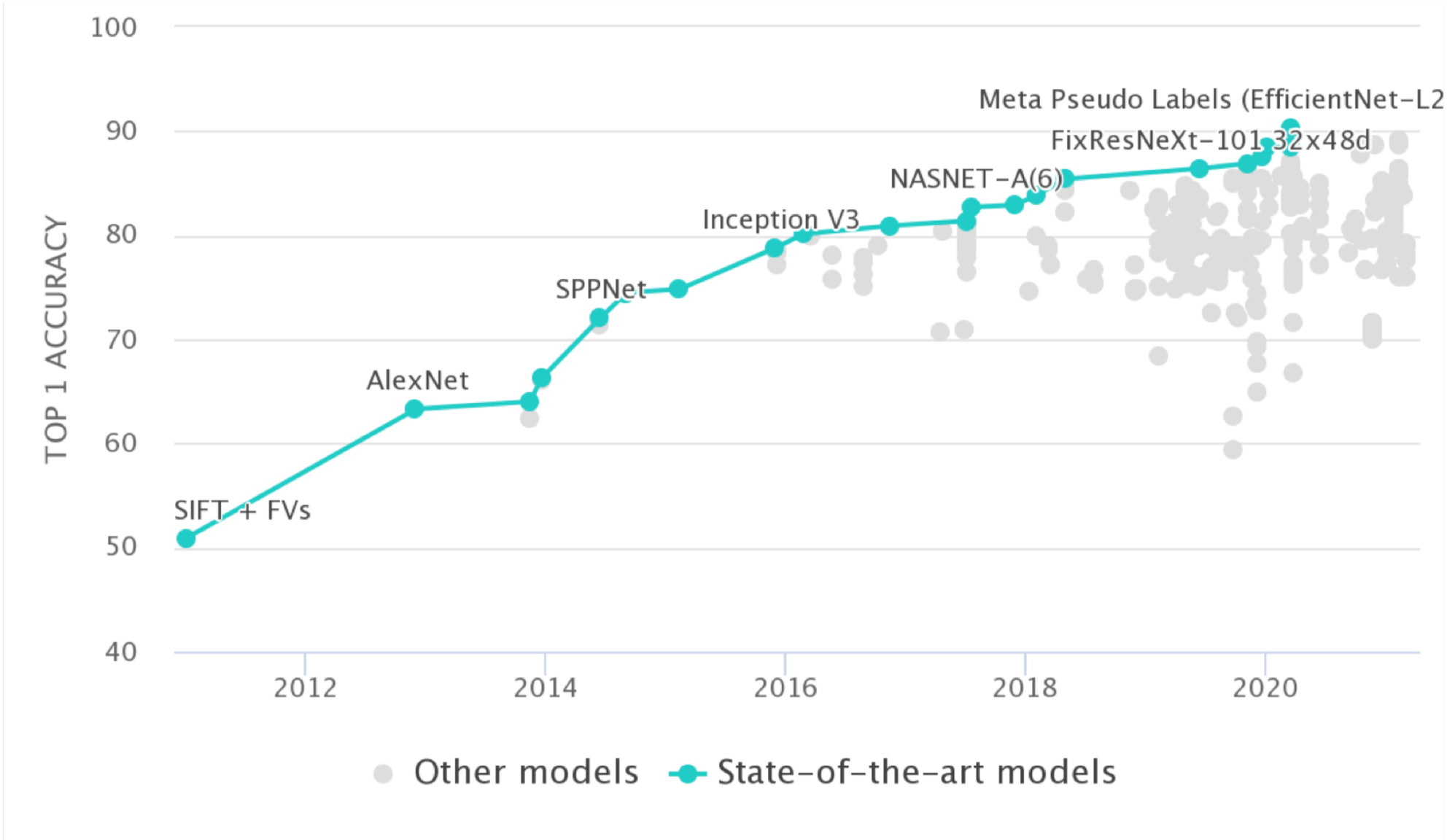
Outline

- **Mon:** Feedforward neural networks
- **Wed:** Neural networks for images - convolutional neural networks (CNNs)
- **Today:**
  1.Transfer Learning
  2.Neural networks for graphs
    - Learning tasks with graphs: examples
    - Graph neural networks
    - Example application: predicting polypharmacy side effects

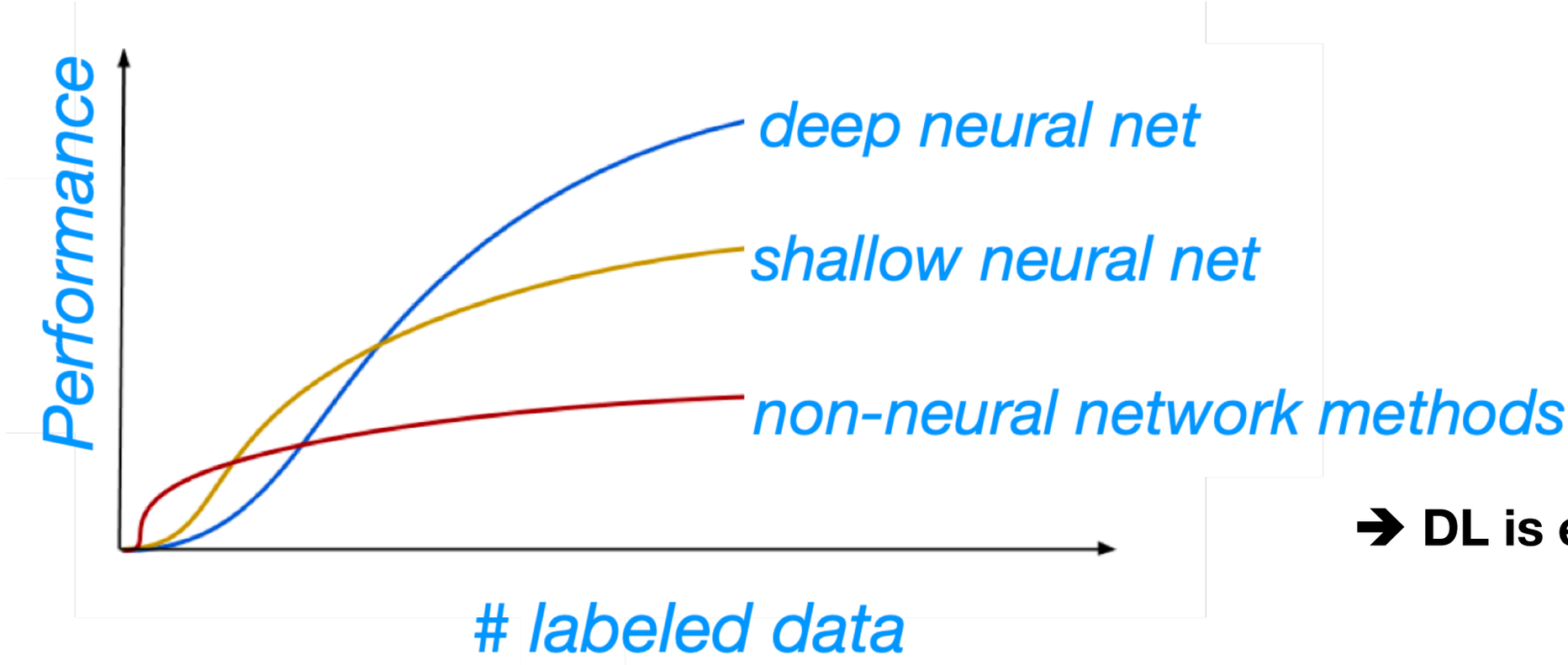
Breakthroughs in deep learning required huge data

IMAGENET

1.2 million ***labeled*** *images*

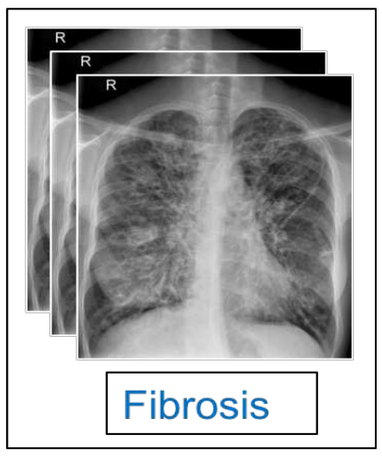


… otherwise…?



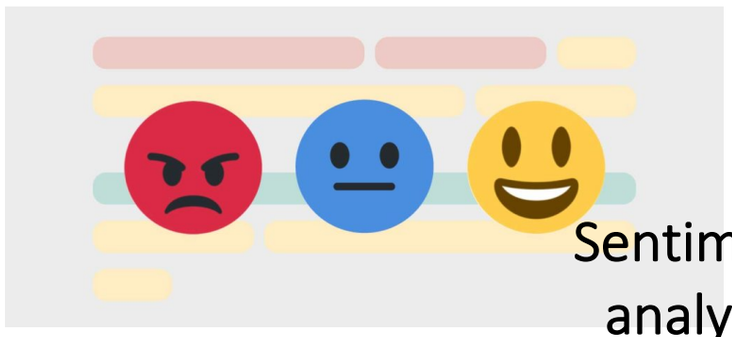
➔ **DL is expensive!**

Adapted from https://www.researchgate.net/figure/Graph-illustrating-the-impact-of-data-available-on-performance-of-traditional-machine_fig1_324457640



Medical imaging

Xie & Richmond, ECCV 2018 workshop

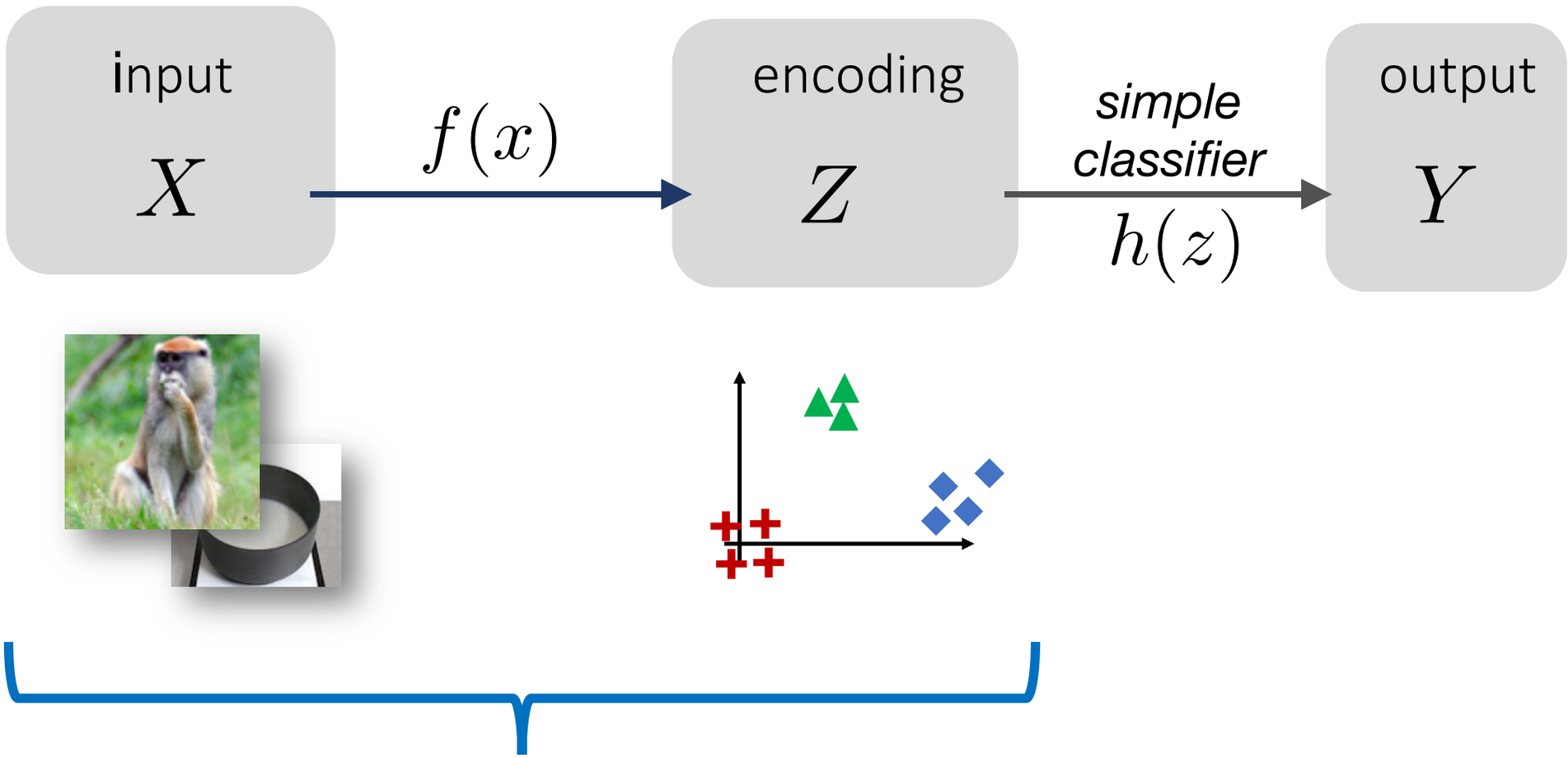


Sentiment analysis

https://hackernoon.com/drafts/523x232ih.png

But what if labels are scarce??

A broad view on deep learning (again)



Idea: Learn encoding from other data!

- With a good representation, a simple, data-efficient classifier can do well ☺

Pre-training

Pre-training

input X → $f(x)$ → encoding Z → $h_{\text{pre}}(z)$ → output W

Pretext task

m data points

Fix & transfer encoding function

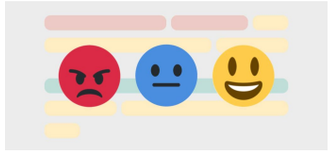
Fine tuning

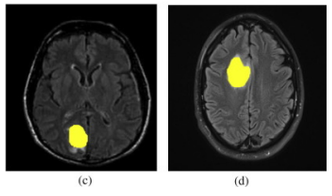
input X → $f(x)$ → encoding Z → $h(z)$ → output Y

Target task

n<<m

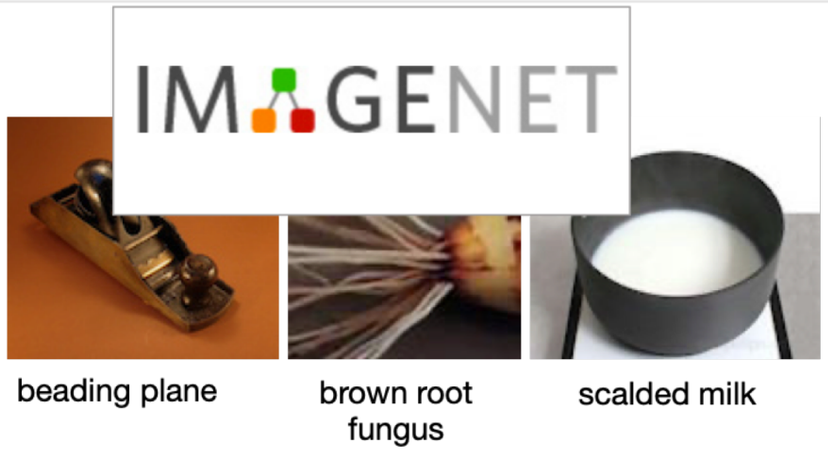
data points





The same pre-trained encoding may work for multiple target tasks

What is the pretext task?

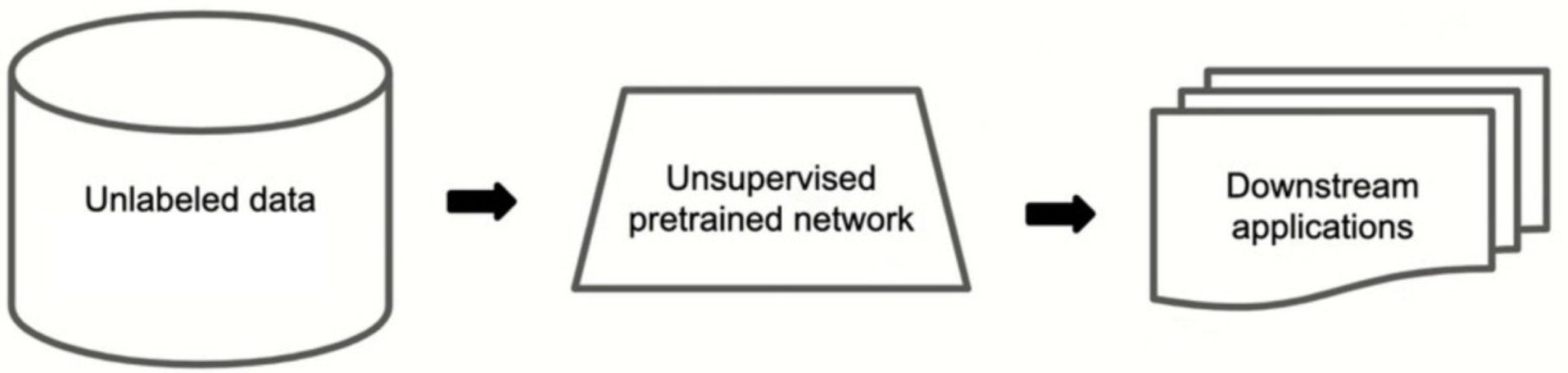


supervised….

Or unsupervised?!

Idea: create labels automatically!

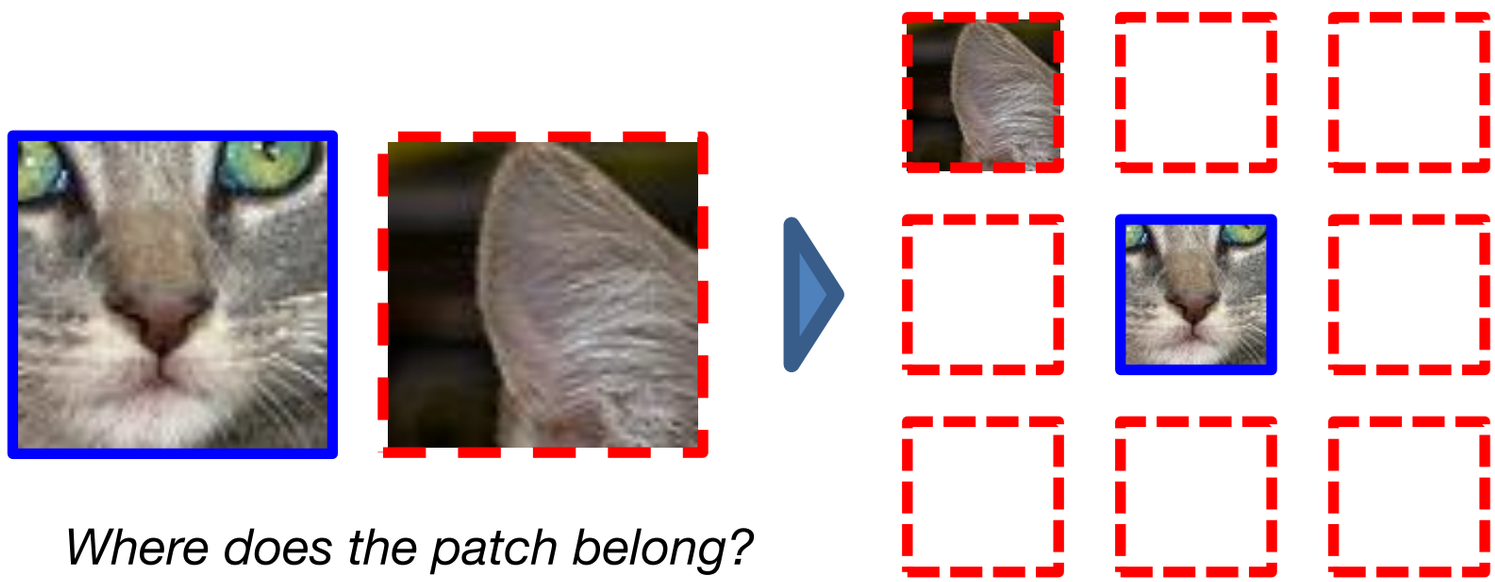
Self-supervised learning



(Image: Ting Chen)

"Fine tuning" is more data- and resource-efficient

Example pretext tasks for images



Where does the patch belong?

(Doersch, Gupta, Efros 2015)

Self-supervision:
labels for the pretext task are generated *automatically*

Target Tasks:
object detection,
visual data mining, …

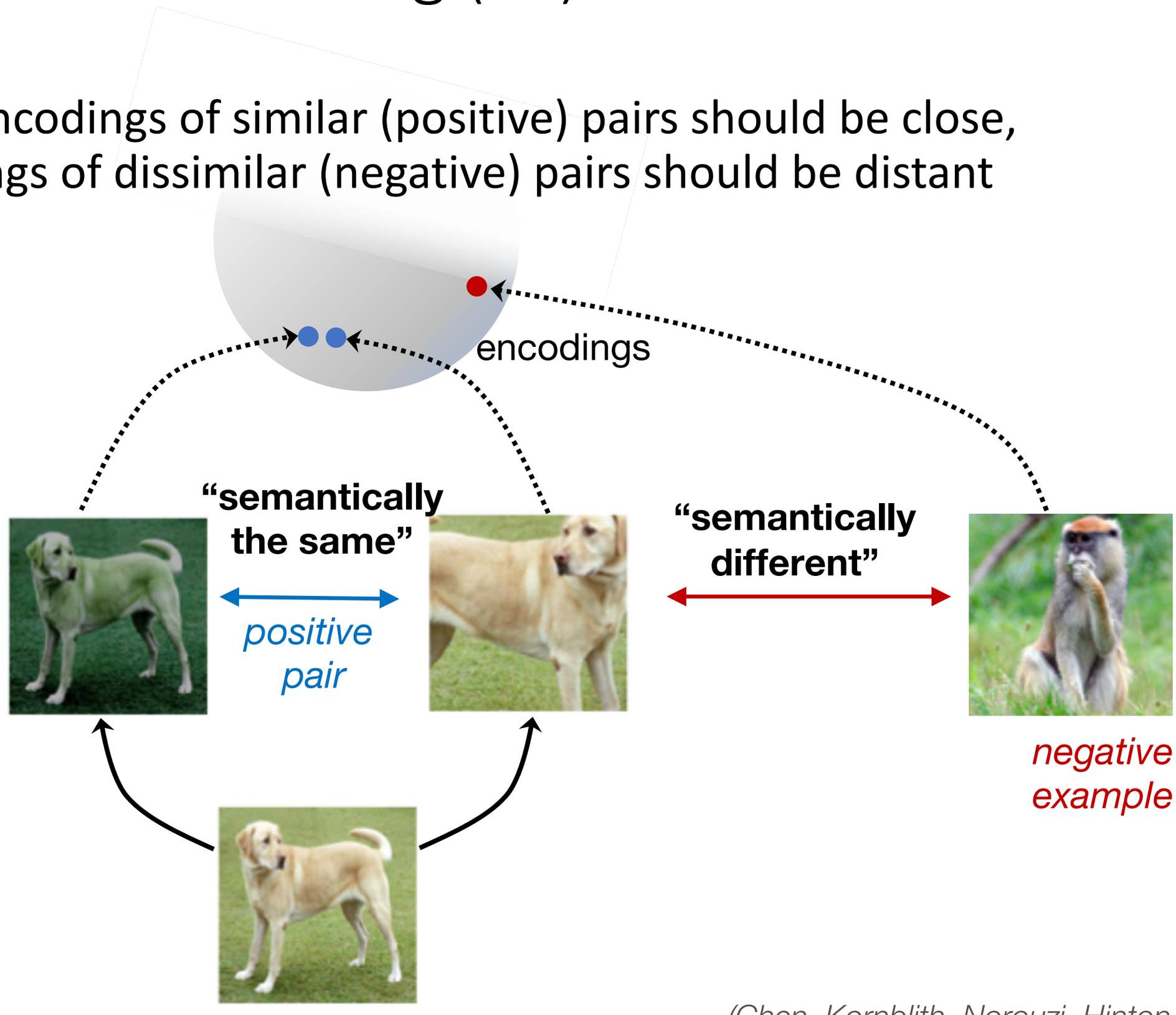
Complete the image?

(Pathak et al. 2016)

(Doersch et al 2015, Noroozi & Favaro 2016, Zhang et al 2016, Pathak et al 2016, Gidaris et al 2018, Chen et al 2019, Kolesnikov et al 2019, …)

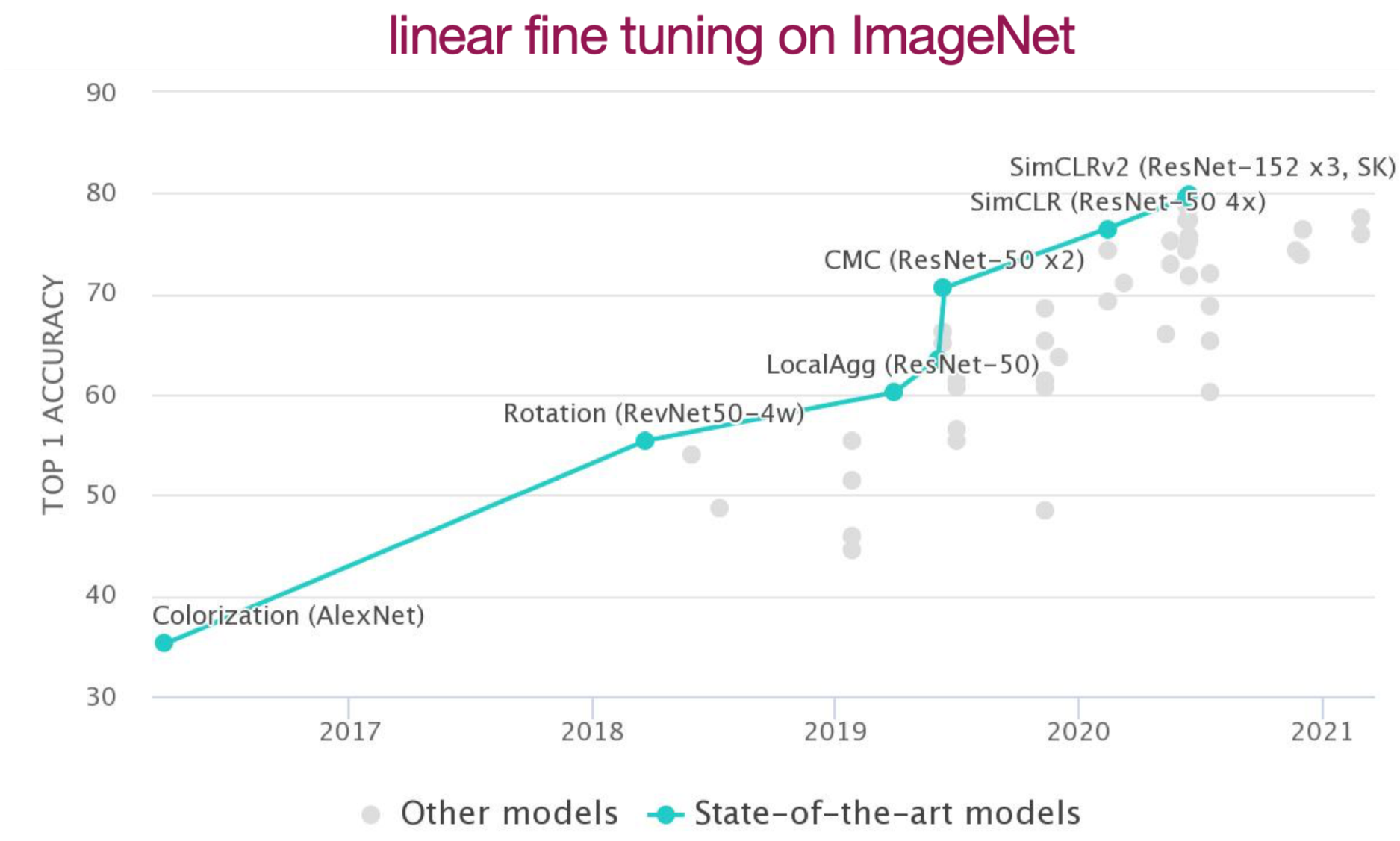
Contrastive learning (CL)

Goal: encodings of similar (positive) pairs should be close, encodings of dissimilar (negative) pairs should be distant

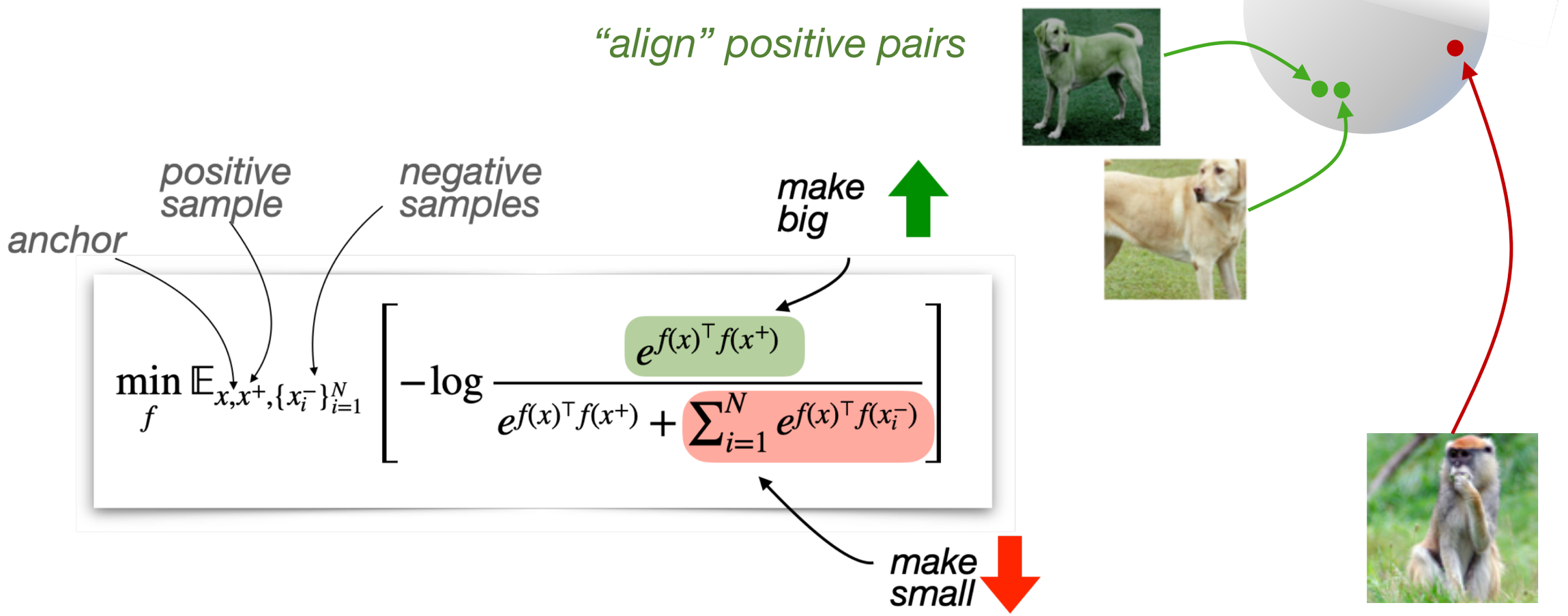


(Chen, Kornblith, Norouzi, Hinton 2020)

Advances of self-supervised learning



CL: Noise contrastive estimation loss

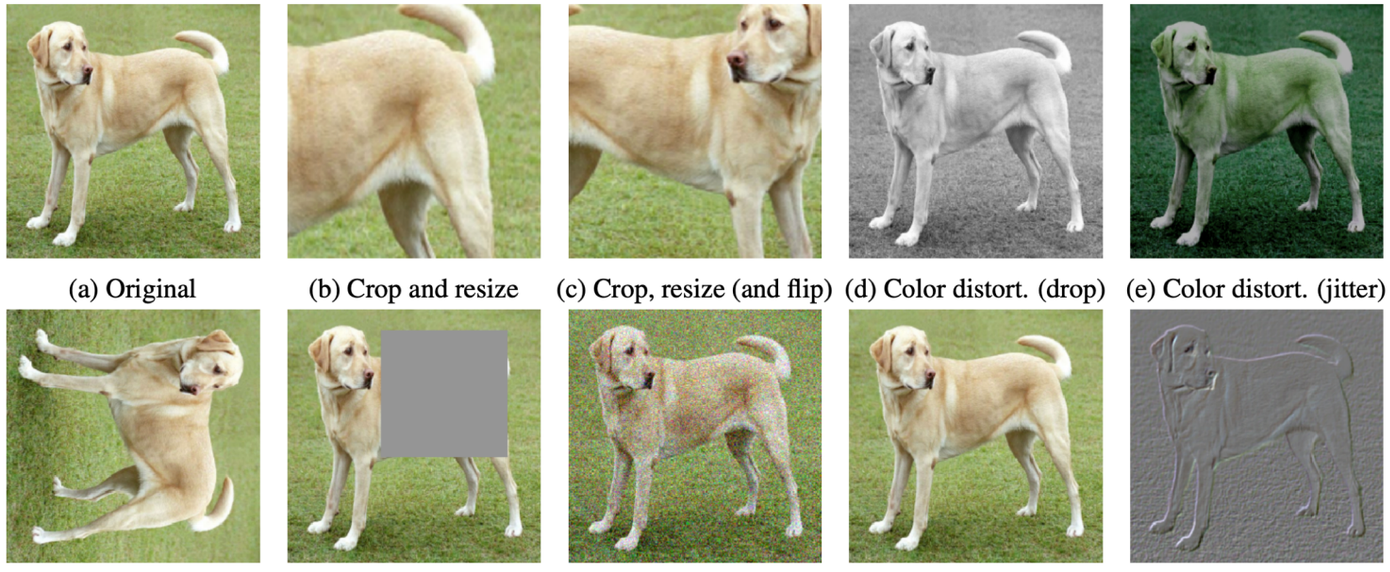


Can outperform supervised pretraining *(He et al 2020, Misra & van der Maaten 2020)*

But: how get positive/negative examples without labels?

Positive and negative examples

Positives:
random combination
of data augmentations

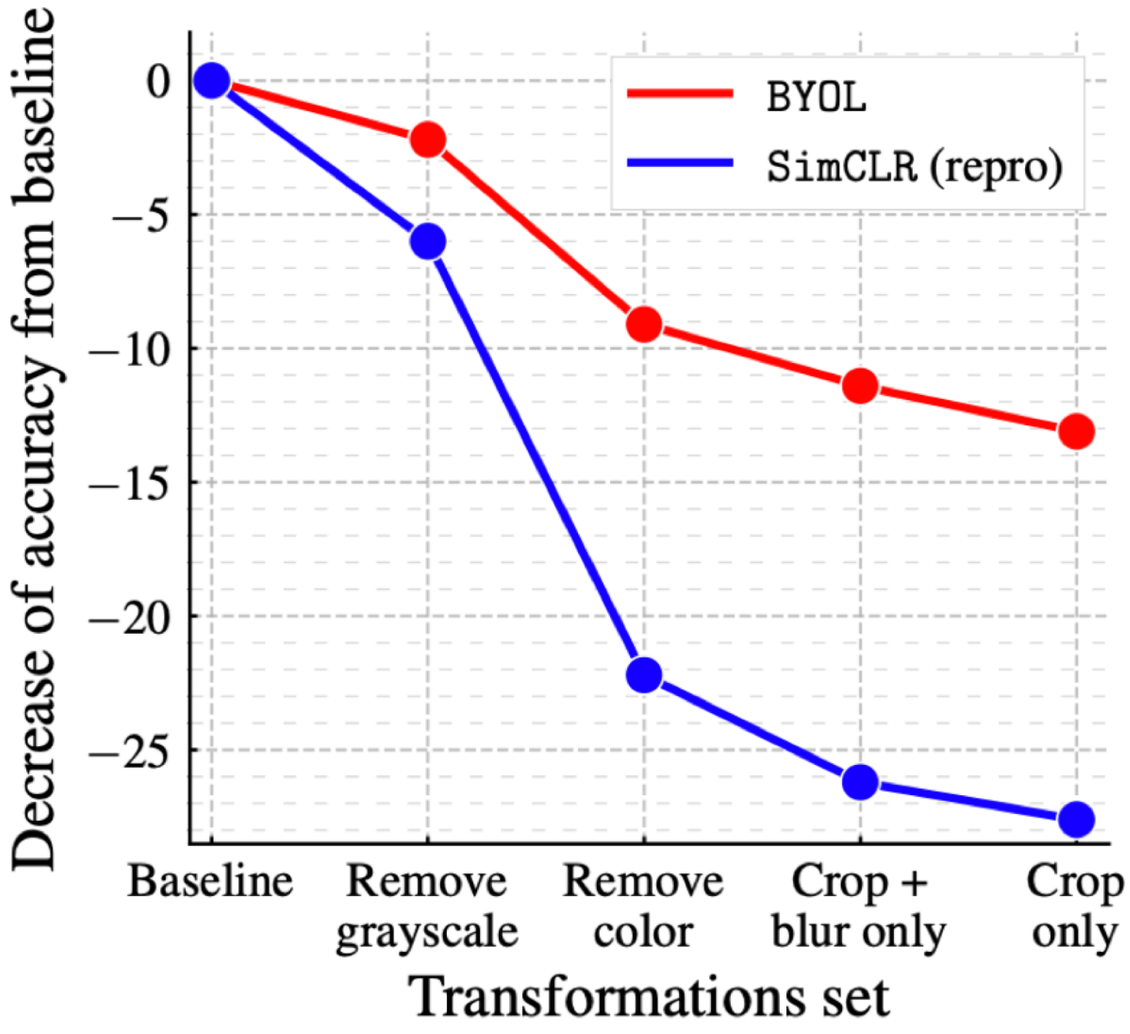


Negatives:
randomly drawn from
dataset

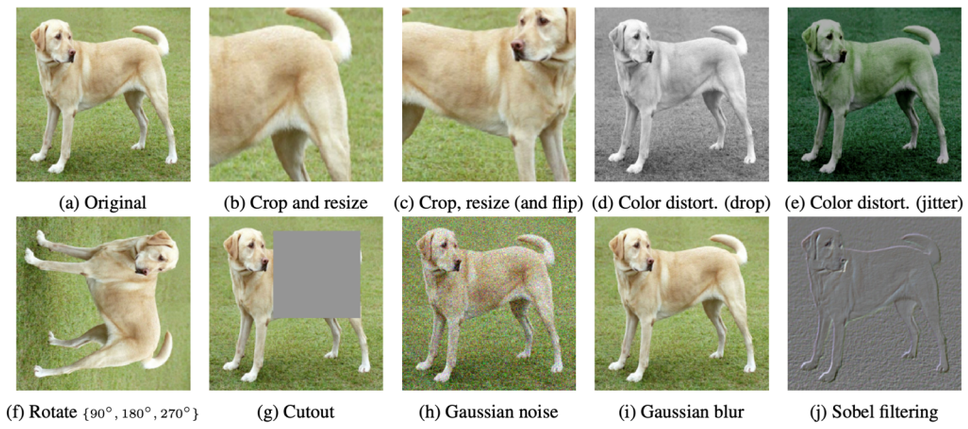
Improvements

- Good positive pairs: lots of data augmentation
- Good negative examples: “hard negative sampling”
- Large batch sizes in SGD
- …

Effect of data augmentation



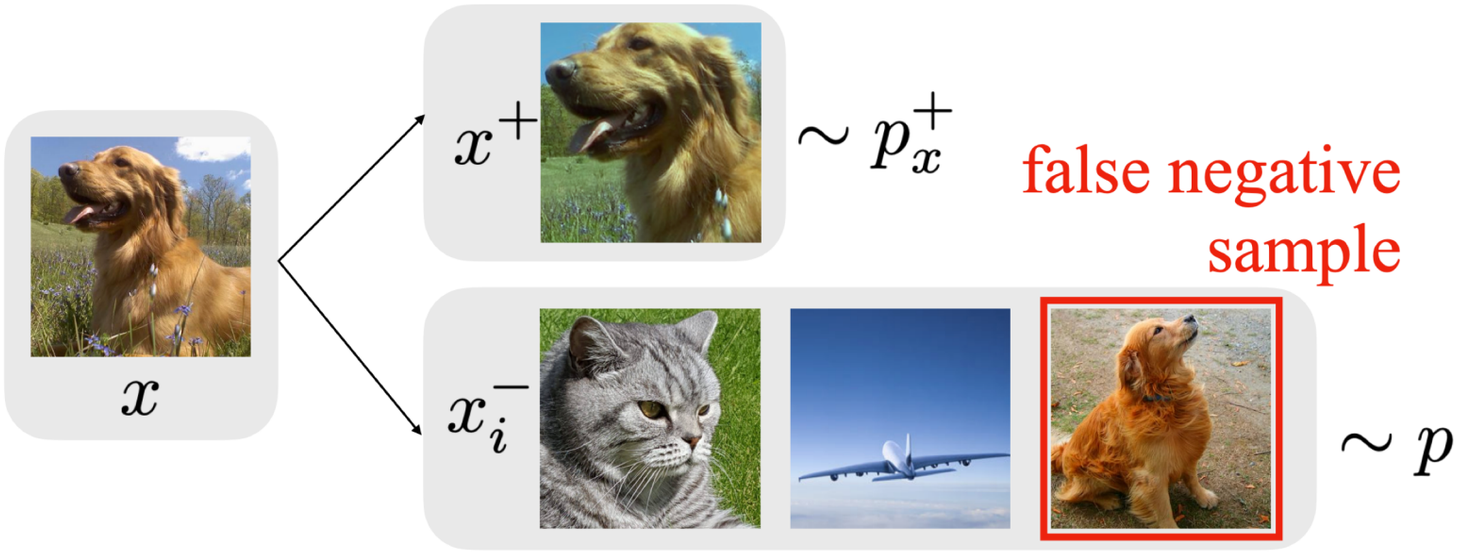
Impact of progressively removing transformations



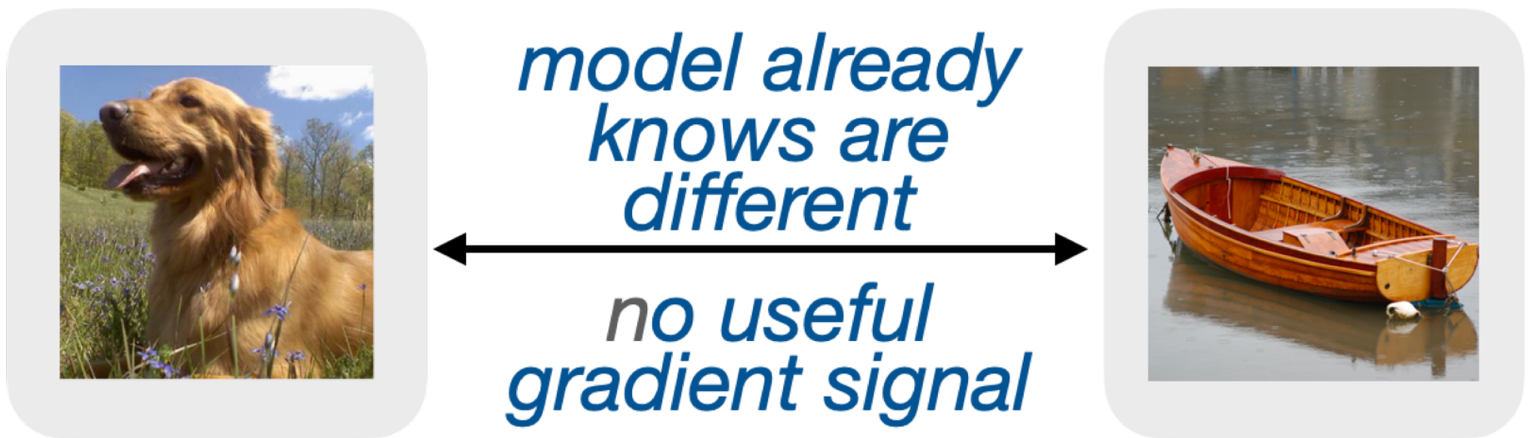
(figure: Grill et al 2020)

More informative negative examples

- False negative examples ➔ debiasing *(Chuang et al 2020)*



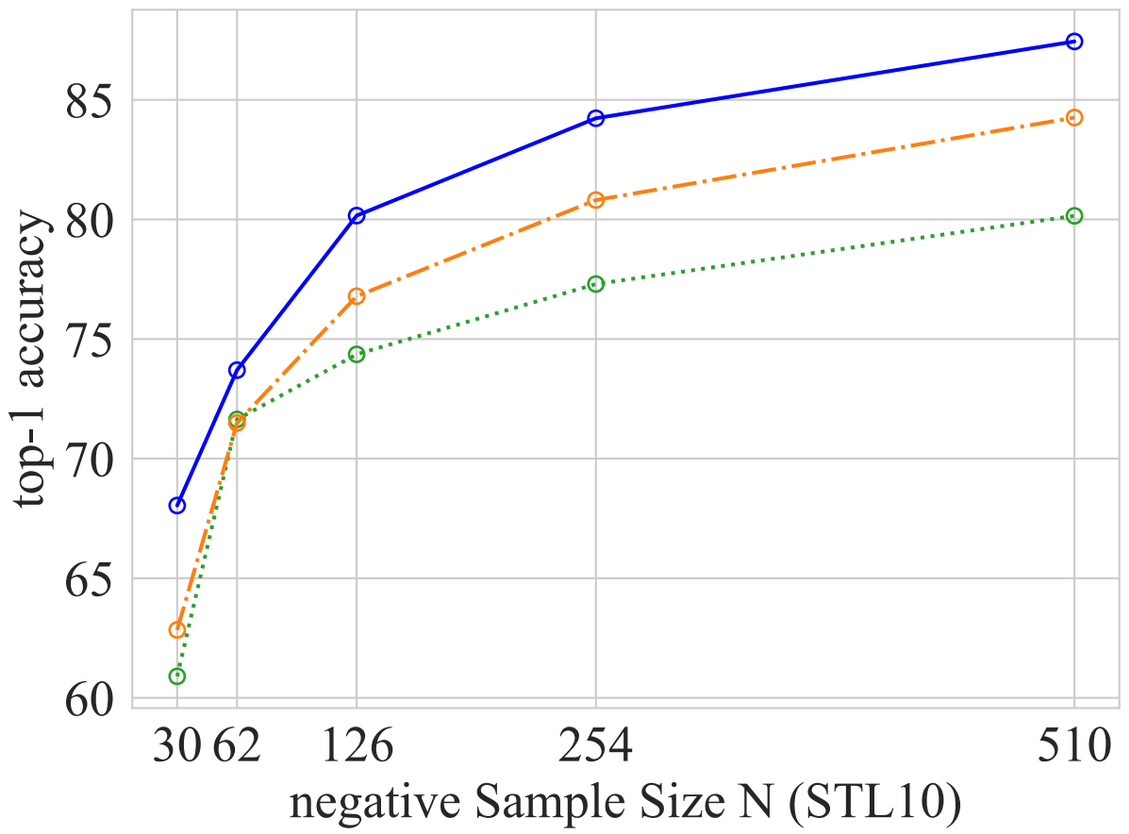
- Uninformative/easy negatives ➔ focus on hard negatives
  (e.g. Robinson et al 2021)



(Figures: C.-Y. Chuang, J. Robinson)

Example hard negatives and effect

Anchor



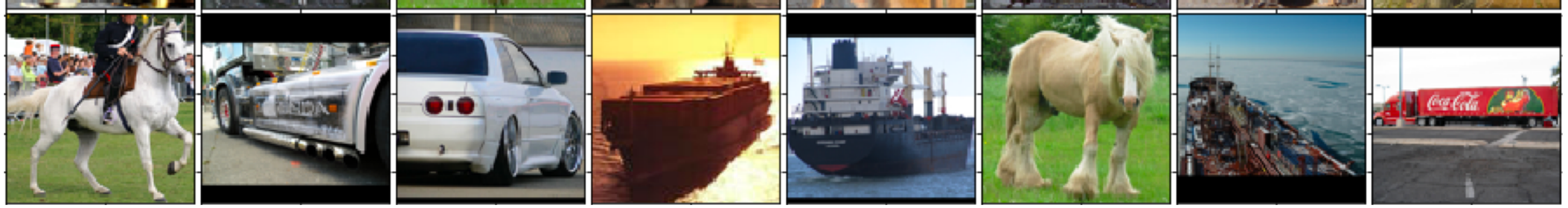
Hard negatives

Debiased CL

Uniform sampling of negatives

Hard negatives

Uniform Negatives

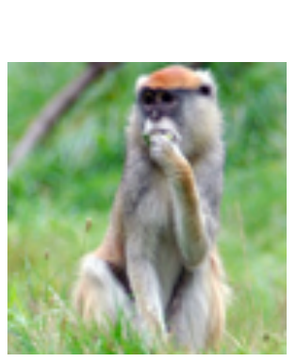


Summary: self-supervised learning

- State of the art for vision and language tasks
- Idea: use auxiliary data to obtain a good data representation
- *Self-supervision*: pretext tasks uses automatically generated label
- Using pre-trained features saves data and computation
- Related to “metric learning”

Model “zoo”:
https://lilianweng.github.io/lil-log/2021/05/31/contrastive-representation-learning.html

Specialized methods to encode data



$\phi(\text{image})$

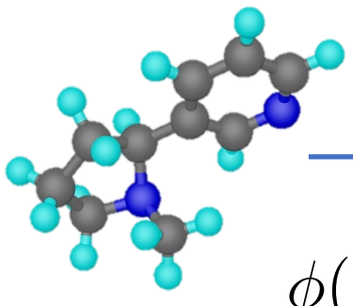
category

CNN (Wed)

RNN

$\phi(\text{text})$

sentiment



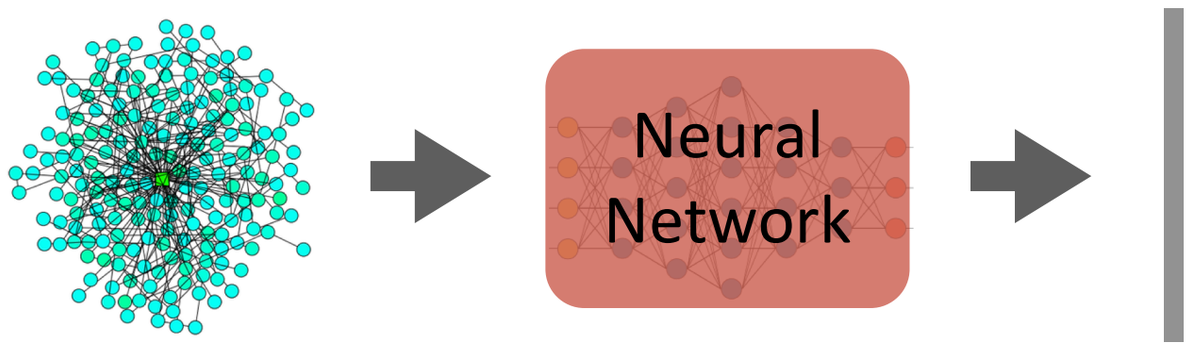
$\phi(\text{molecule})$

property

GNN (today)

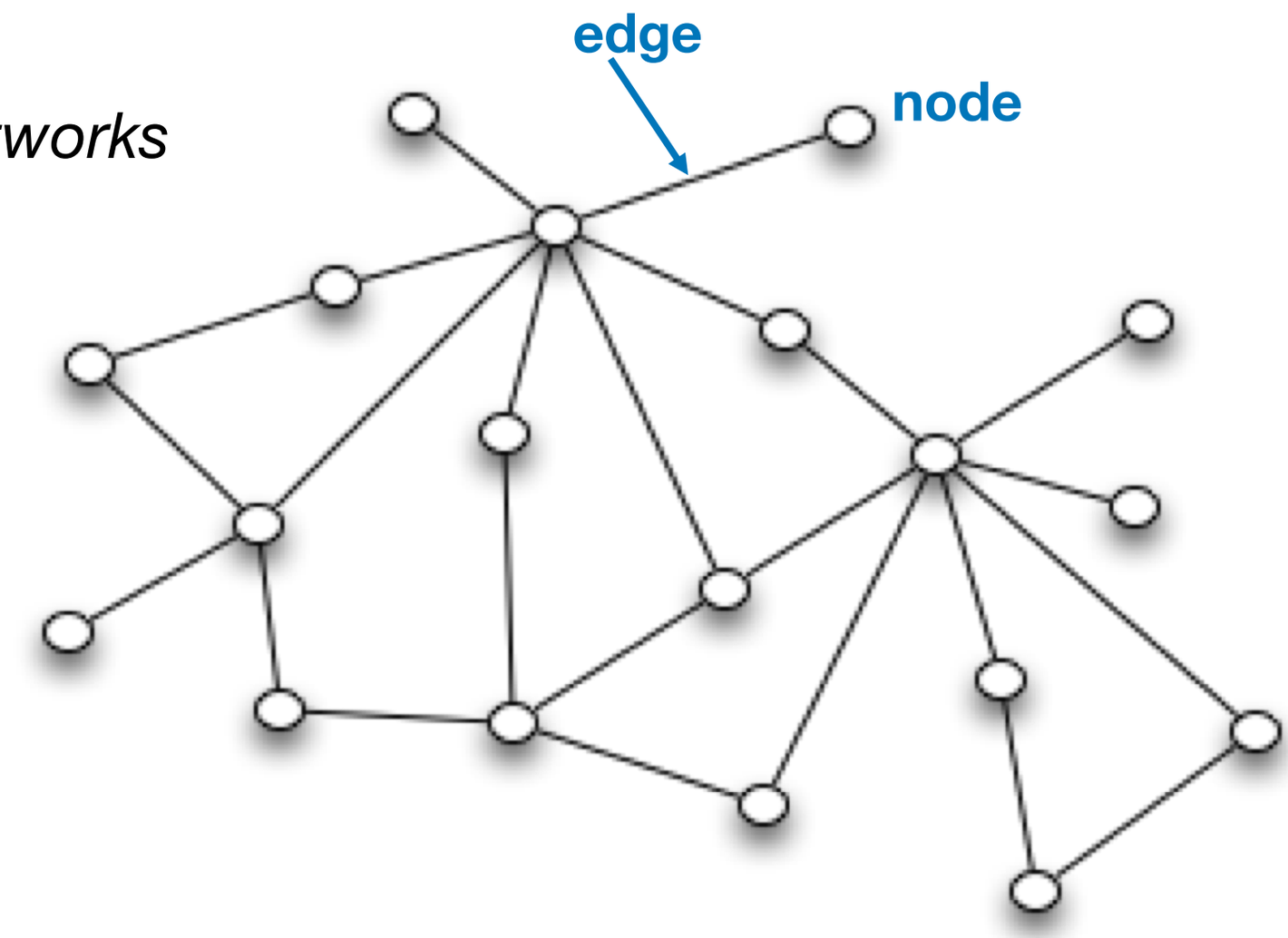
Outline

- Learning tasks with graphs: examples
- Graph neural networks
- Example application:
  predicting polypharmacy side effects

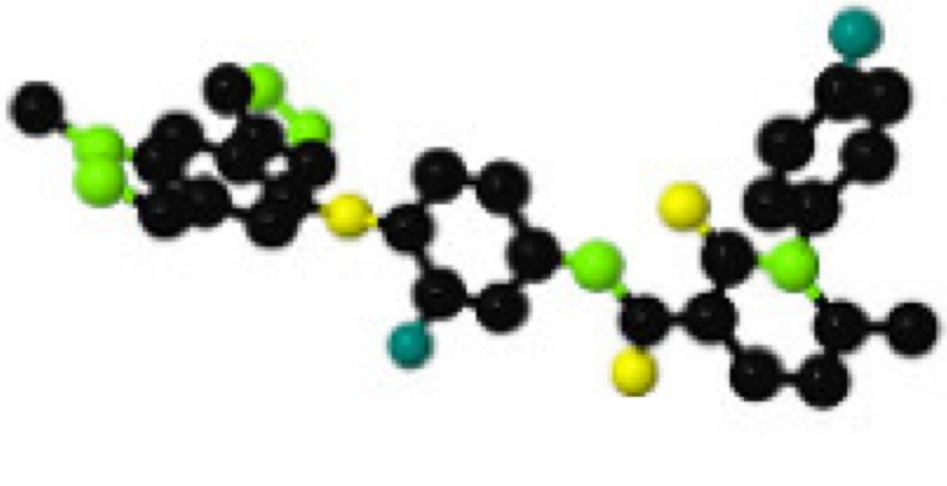


Networks / Graphs

- nodes and connections (edges)
- possibly side information (attributes) for each node
- many examples:
  - ✦ *social networks*
  - ✦ *traffic networks*
  - ✦ *protein interaction networks*
  - ✦ *climate networks*
  - ✦ *brain*
  - ✦ *citation networks*
  - ✦ *internet …*



Example: molecule property prediction



Predict a label for the entire graph

Graph
Molecule

→

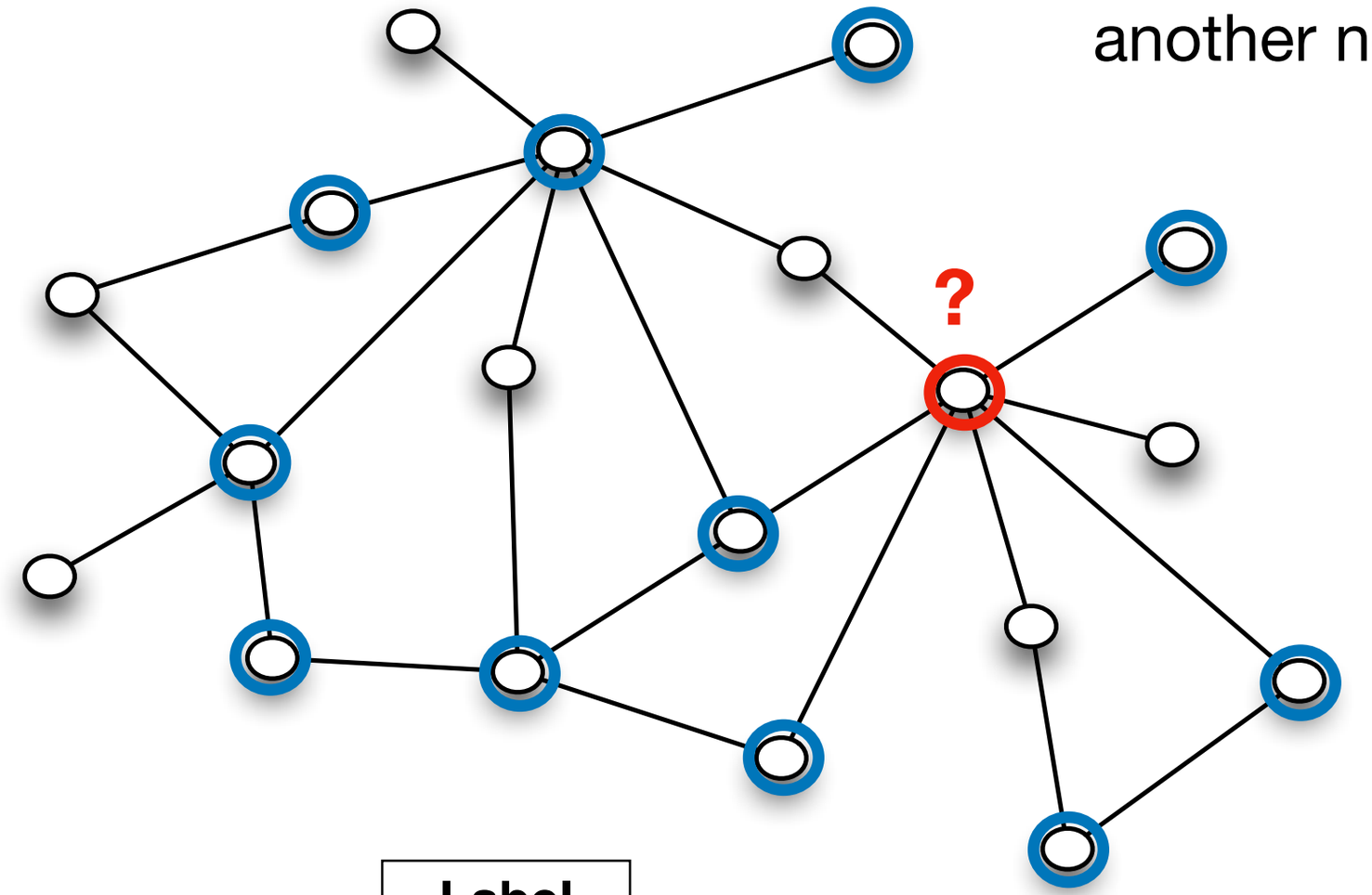
Property
Solubility
Toxicity
Drug efficacy
…

(Duvenaud et al, 2015, Stokes et al 2020,…)

On the cover: Antibiotic resistance is a pervasive public health problem, requiring the adoption of creative approaches to drug discovery. In this issue, ... Show more

Example: social network

Observe some nodes, predict a label for another node



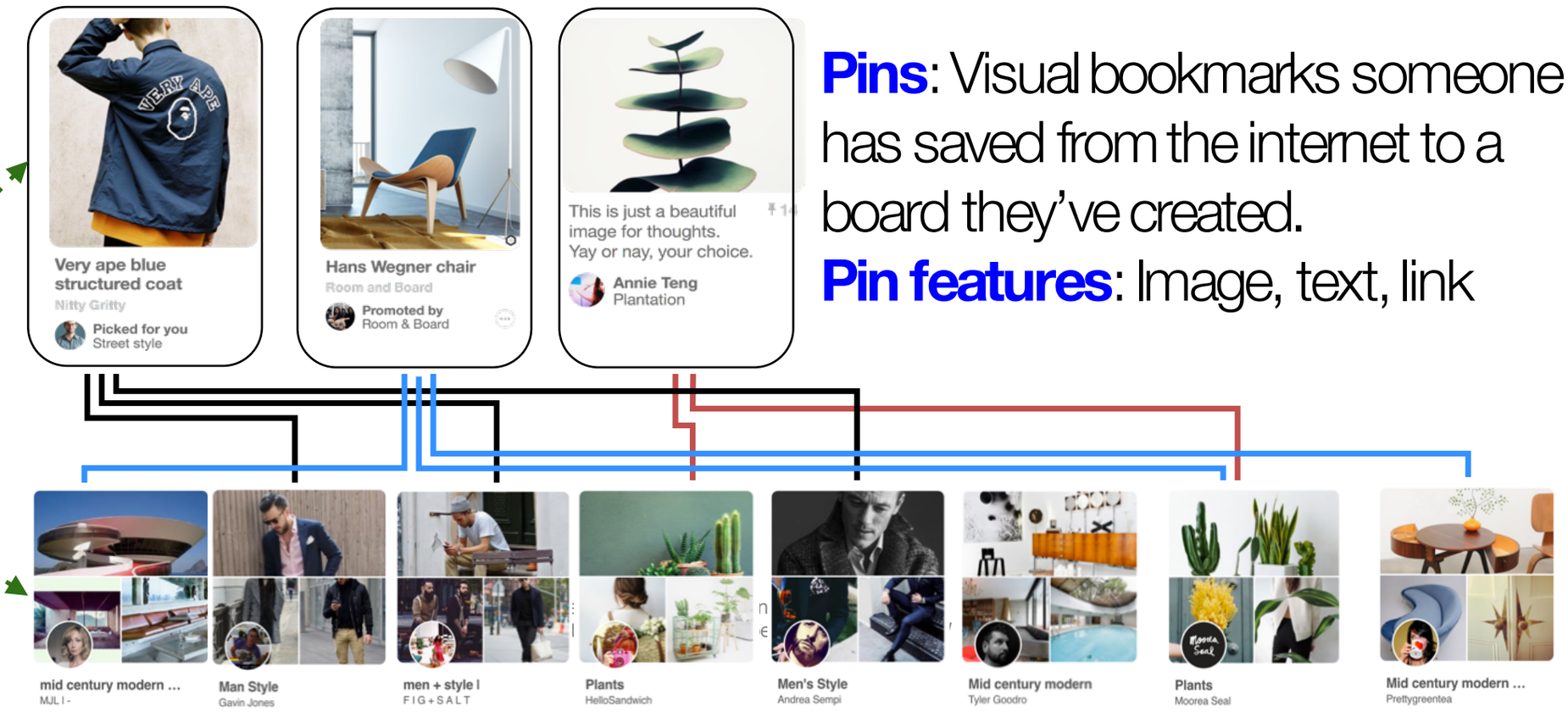
Node
Person

Label
job
interests
…

Example: recommender system

Given a node, predict other (similar) nodes

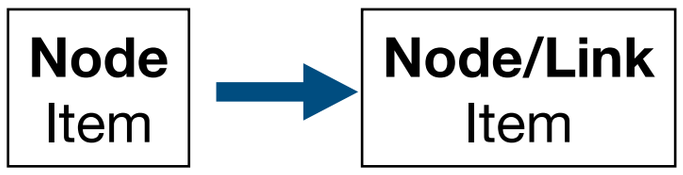
2 types of nodes



Pins: Visual bookmarks someone has saved from the internet to a board they've created.

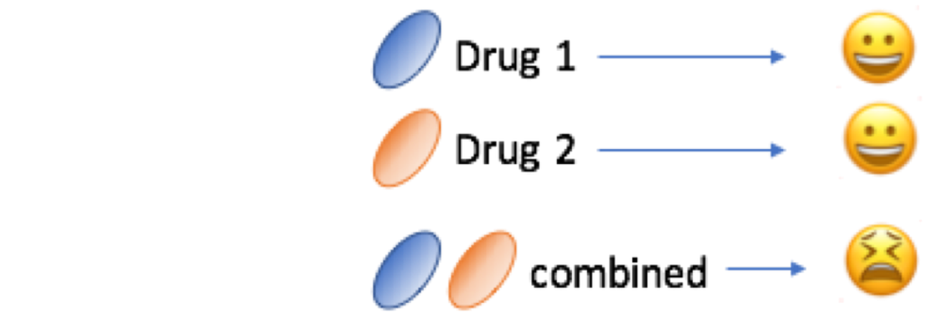
Pin features: Image, text, link

Boards

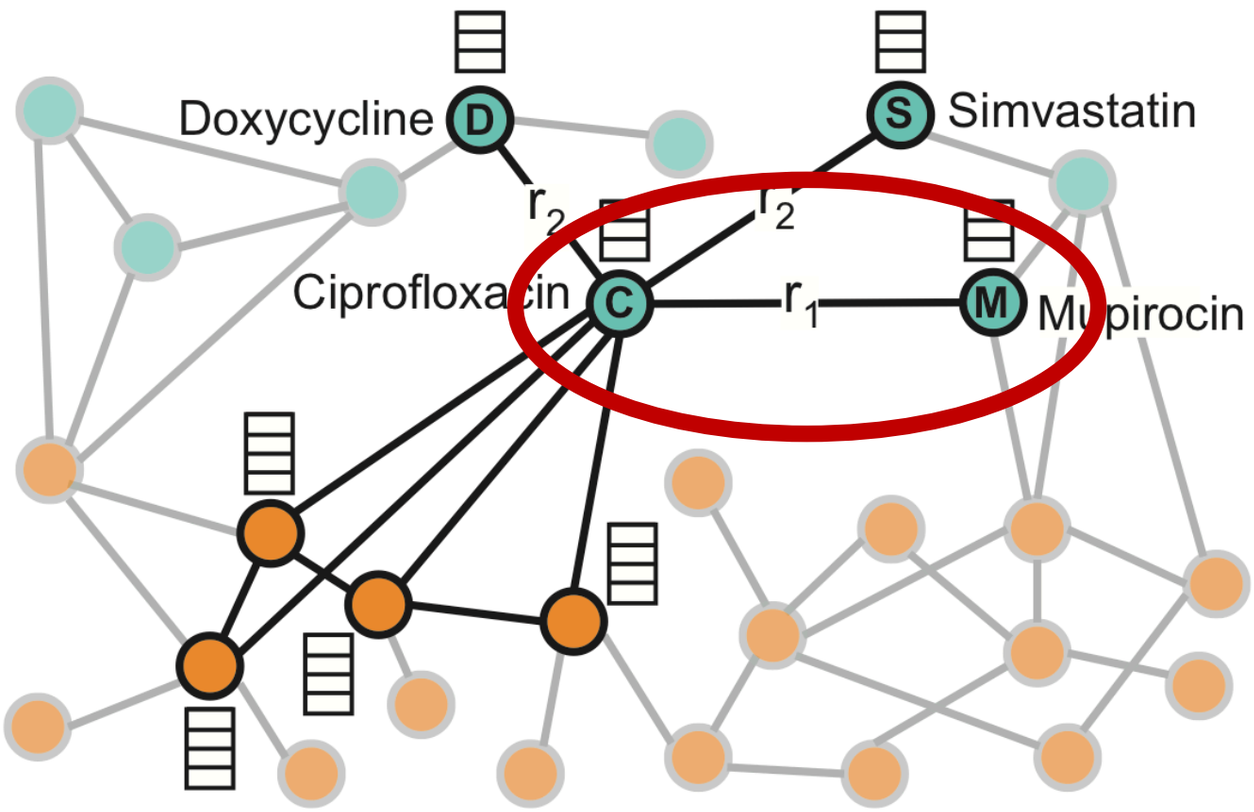


(Ying et al, 2018; illustration: J. Leskovec)

Example: Polypharmacy side effects



Predict if there is an edge between two nodes



Drugs

Proteins

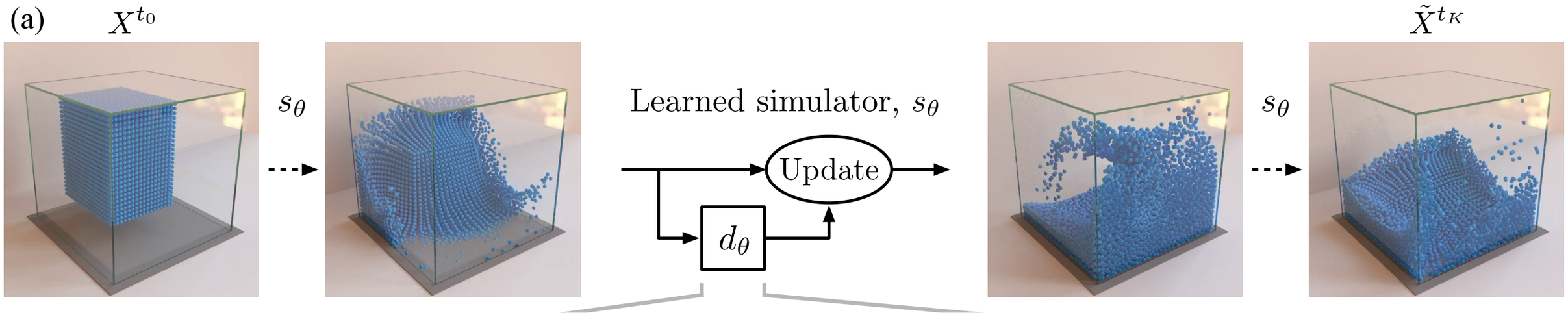
Pair of Nodes
Drugs

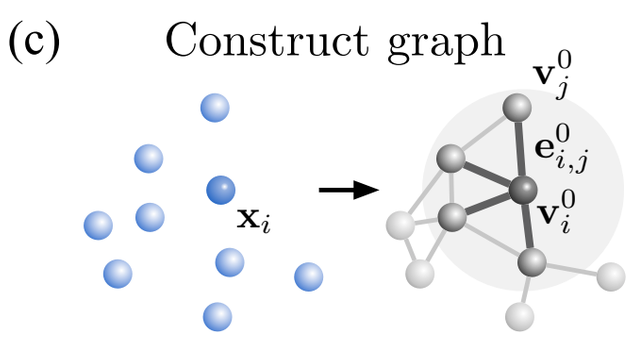
→

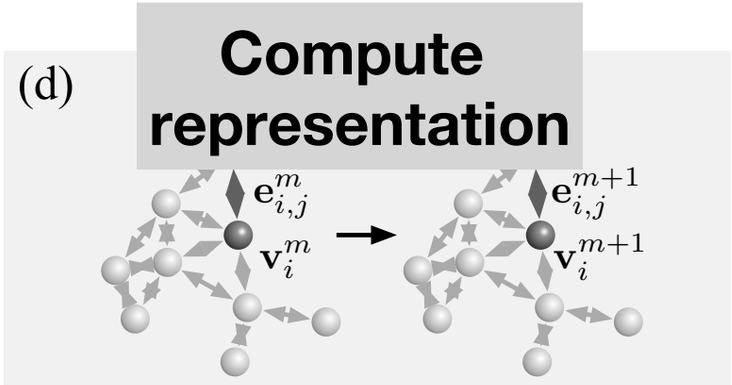
Edge
Interaction type

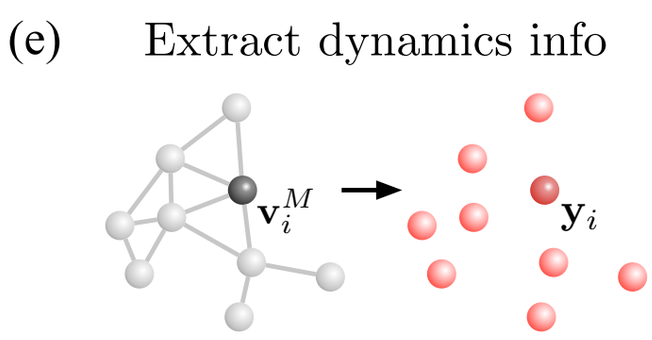
(Zitnik et al, 2018)

Learning to Simulate Physics









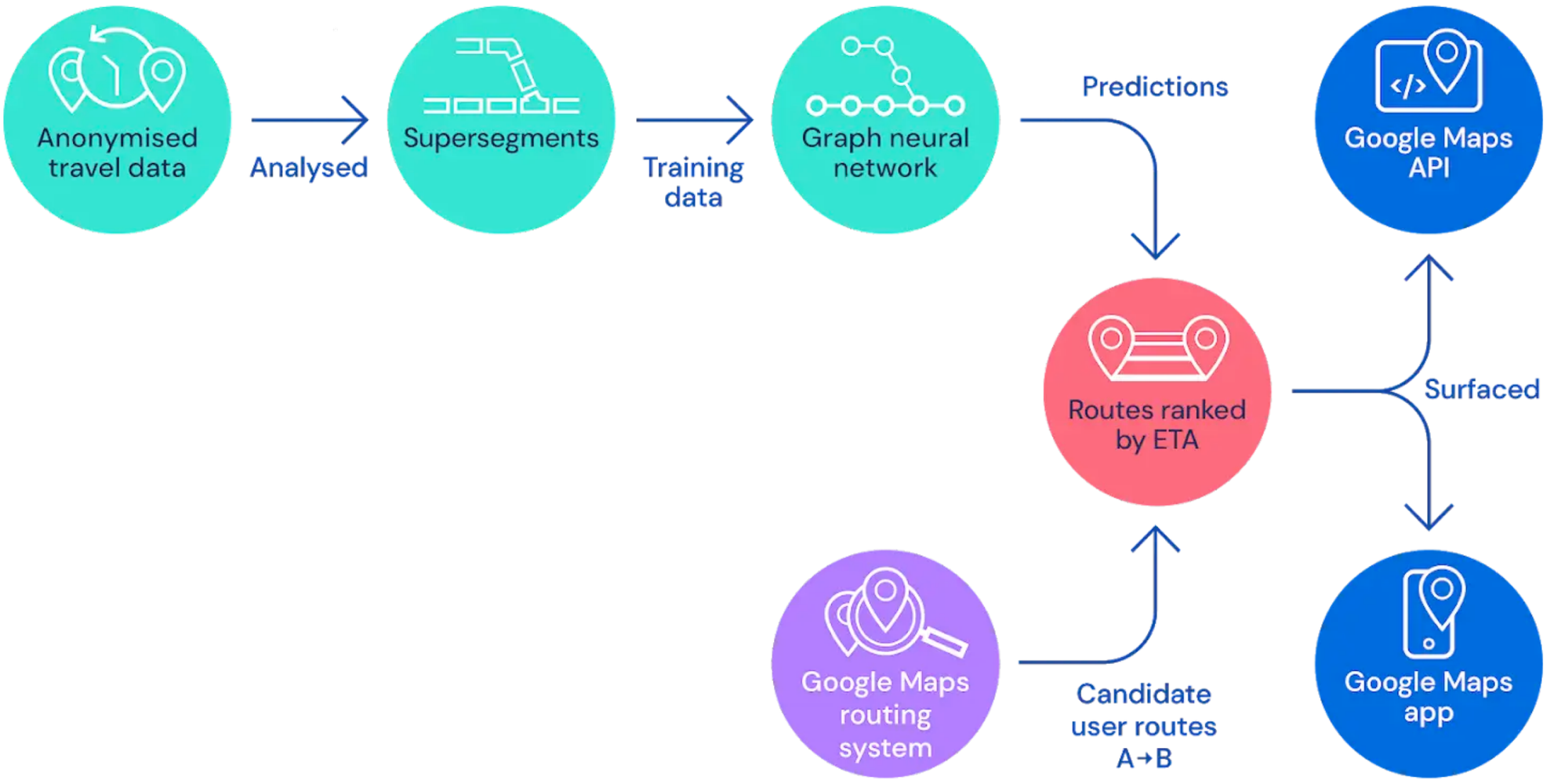
Collection of Nodes
Particles

→

New state

(Sanchez-Gonzalez et al, 2020)

Predicting traffic times



https://deepmind.com/blog/article/traffic-prediction-with-advanced-graph-neural-networks

And more …

Web Image Search Gets Better With Graph Neural Networks

A new approach to image search uses images returned by traditional search methods as nodes in a graph neural network through which similarity signals are propagated, achieving improved ranking in cross-modal retrieval.

Machine Learning

Irene Maxwell from NAVER LABS Europe FOLLOW
01 Dec, 2020

Traffic forecasting for every city?

BY GCN STAFF | NOV 23, 2020

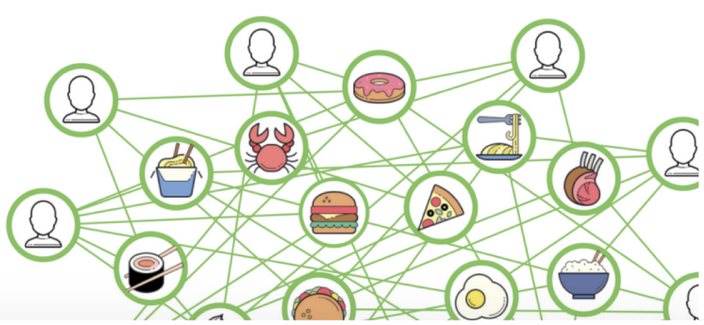
Researchers from Argonne National Laboratory are using a graph neural network and a new transfer learning approach to more accurately forecast large-scale traffic patterns.

Uber Engineering

AI General Engineering

Food Discovery with Uber Eats: Using Graph Learning to Power Recommendations

Ankit Jain, Isaac Liu, Ankur Sarda, and Piero Molino



The next big thing: the use of graph neural networks to discover particles

September 24, 2020 | Zack Savitsky

Share Tweet Email

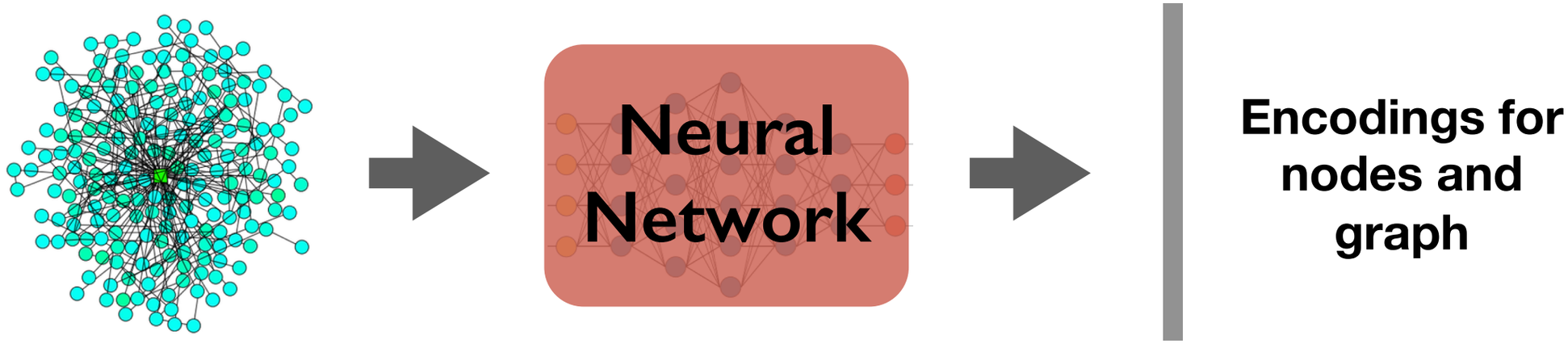
Machine learning algorithms can beat the world's hardest video games in minutes and solve complex equations faster than the collective efforts of generations of physicists. But the conventional algorithms still struggle to pick out stop signs on a busy street.

Object identification continues to hamper the field of machine learning — especially when the pictures are multidimensional and complicated, like the ones particle detectors take of collisions in high-energy physics experiments. However, a new class of neural networks is helping these models boost their pattern recognition abilities, and the technology may soon be implemented in particle physics experiments to optimize data analysis.

This summer, Fermilab physicists made an advance in their effort to embed graph neural networks into the experimental systems. Scientist Lindsey Gray updated software that allows these cutting-edge algorithms to be deployed on data from the Large Hadron Collider at CERN. For the first time, these networks will be integrated into particle physics experiments to process detector data directly — opening the flood gates for a major jump in efficiency that will yield more precise insight from current and future detectors.

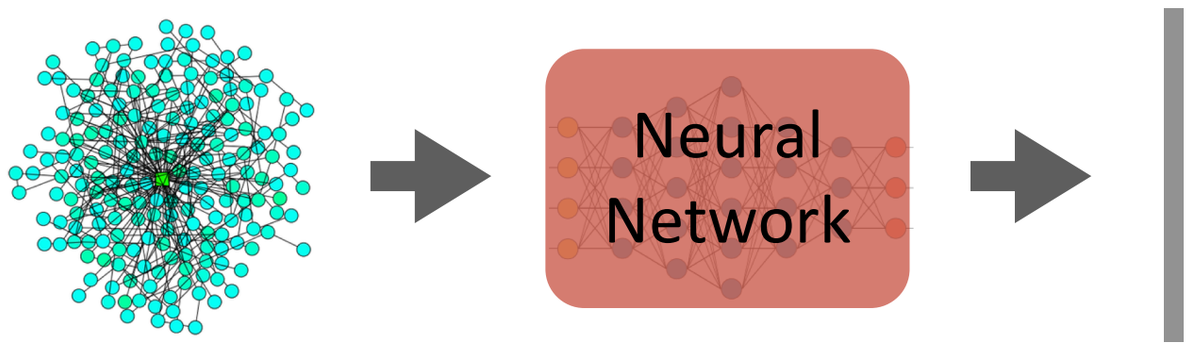
Neural Networks for graphs?

- Need a good representation of nodes or of the entire graph
- Capture node features and neighborhood relations
- Graph Neural Networks (GNNs)

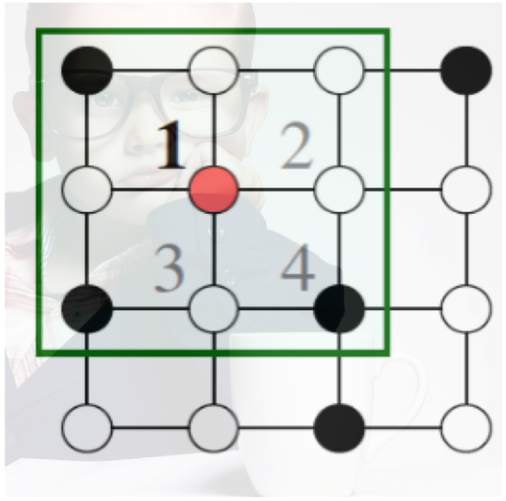


Outline

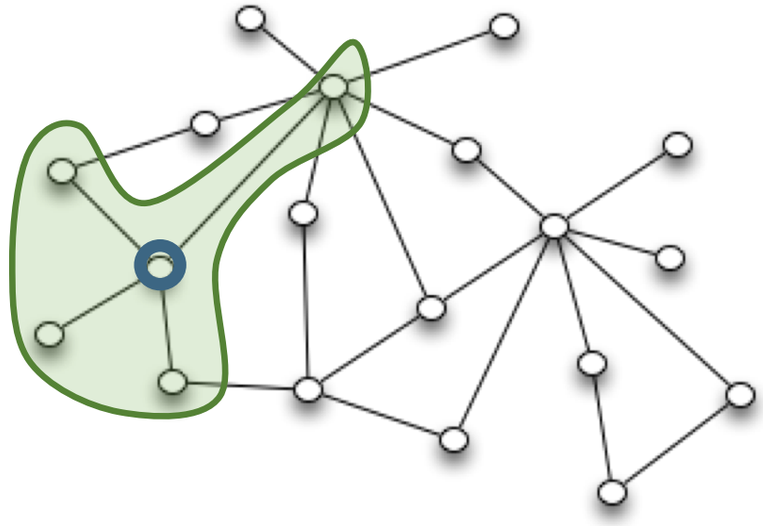
- Learning tasks with graphs: examples
- Graph neural networks
- Example application:
  predicting polypharmacy side effects



Recall CNNs…

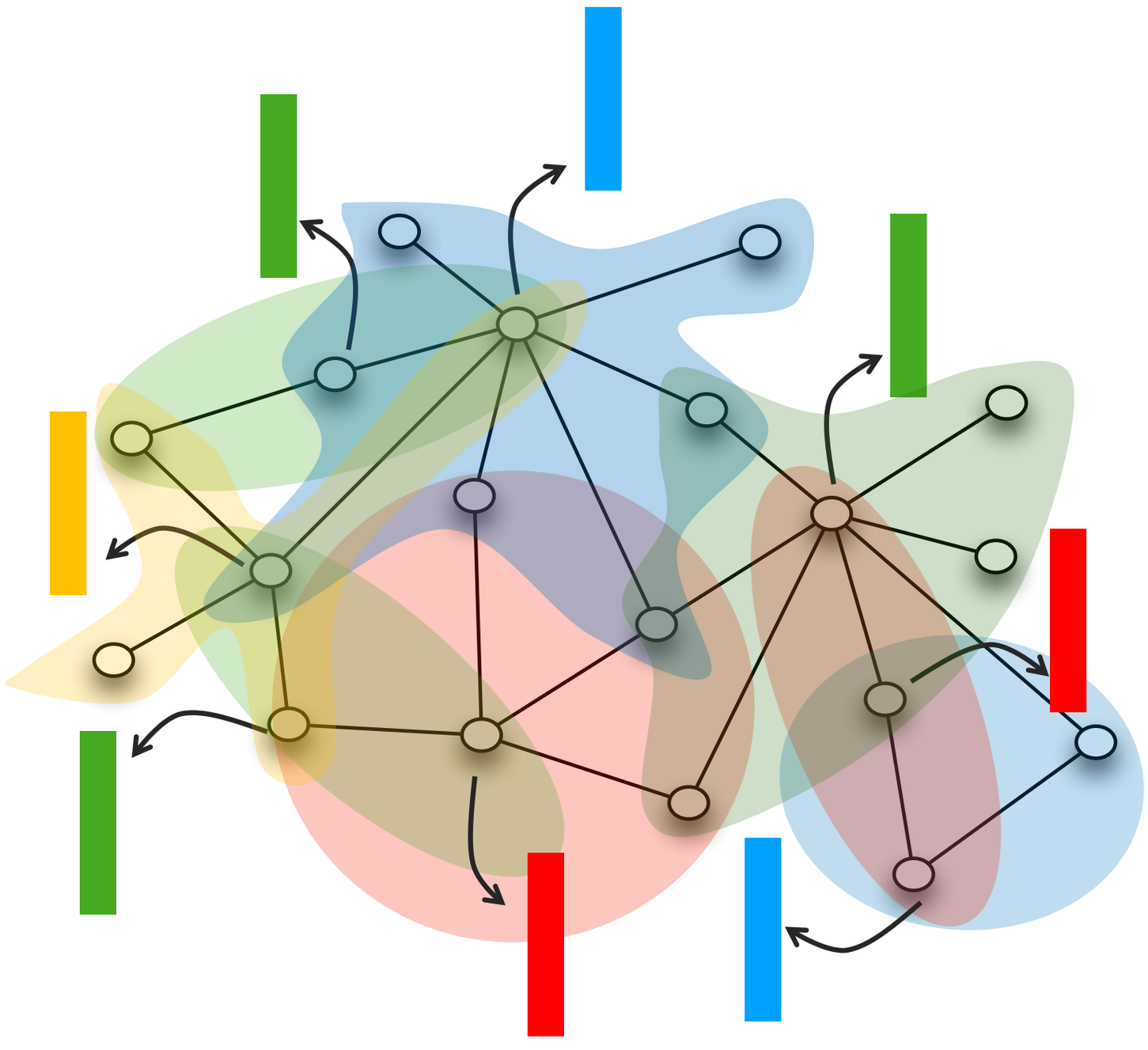


Encode local neighborhoods (patches)



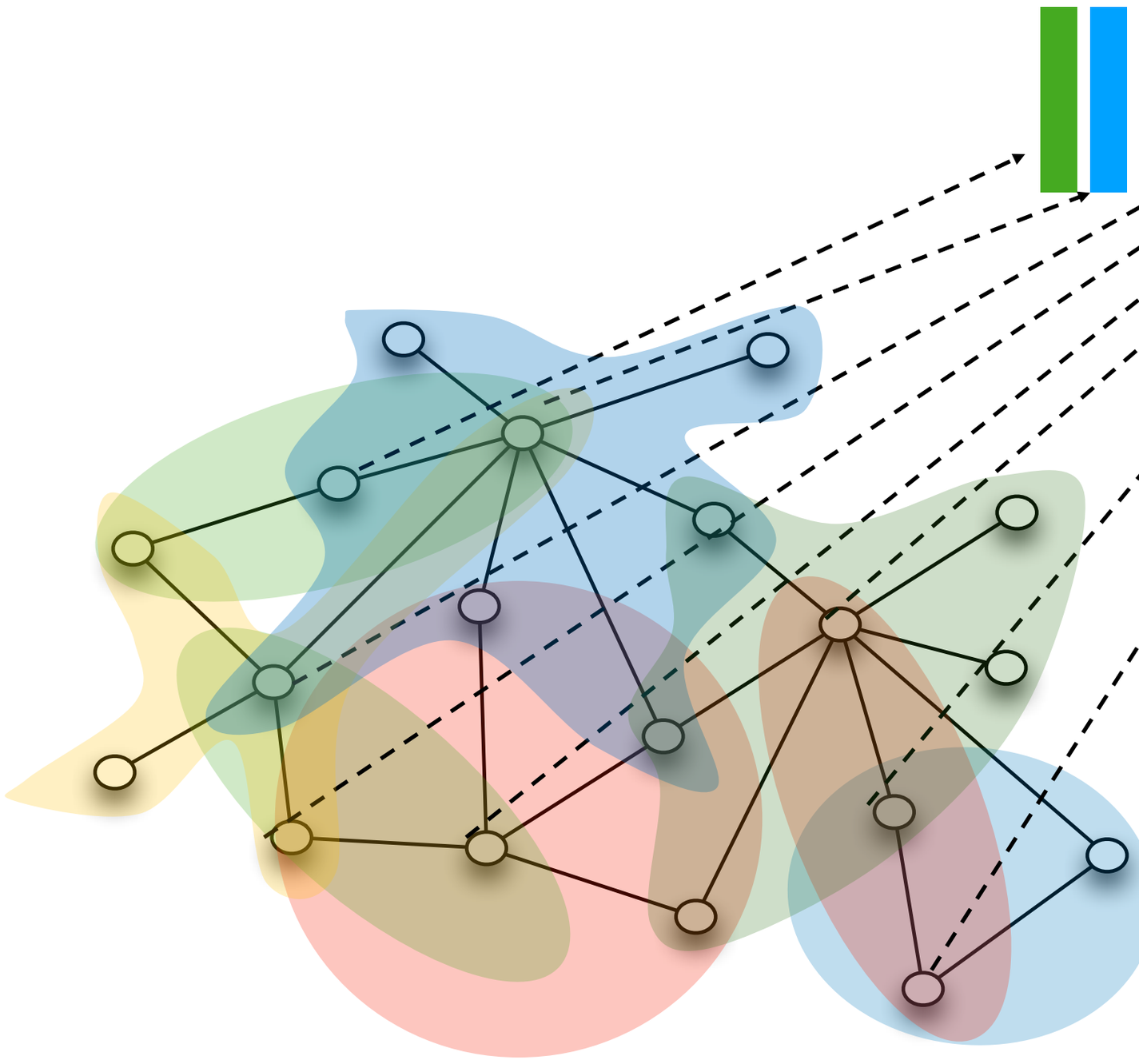
Encode local neighborhoods (nodes that can be reached within *K* hops)

Big picture of Graph Neural Networks (GNNs)



1. Node embeddings:
encode the local neighborhood of each node separately via aggregations

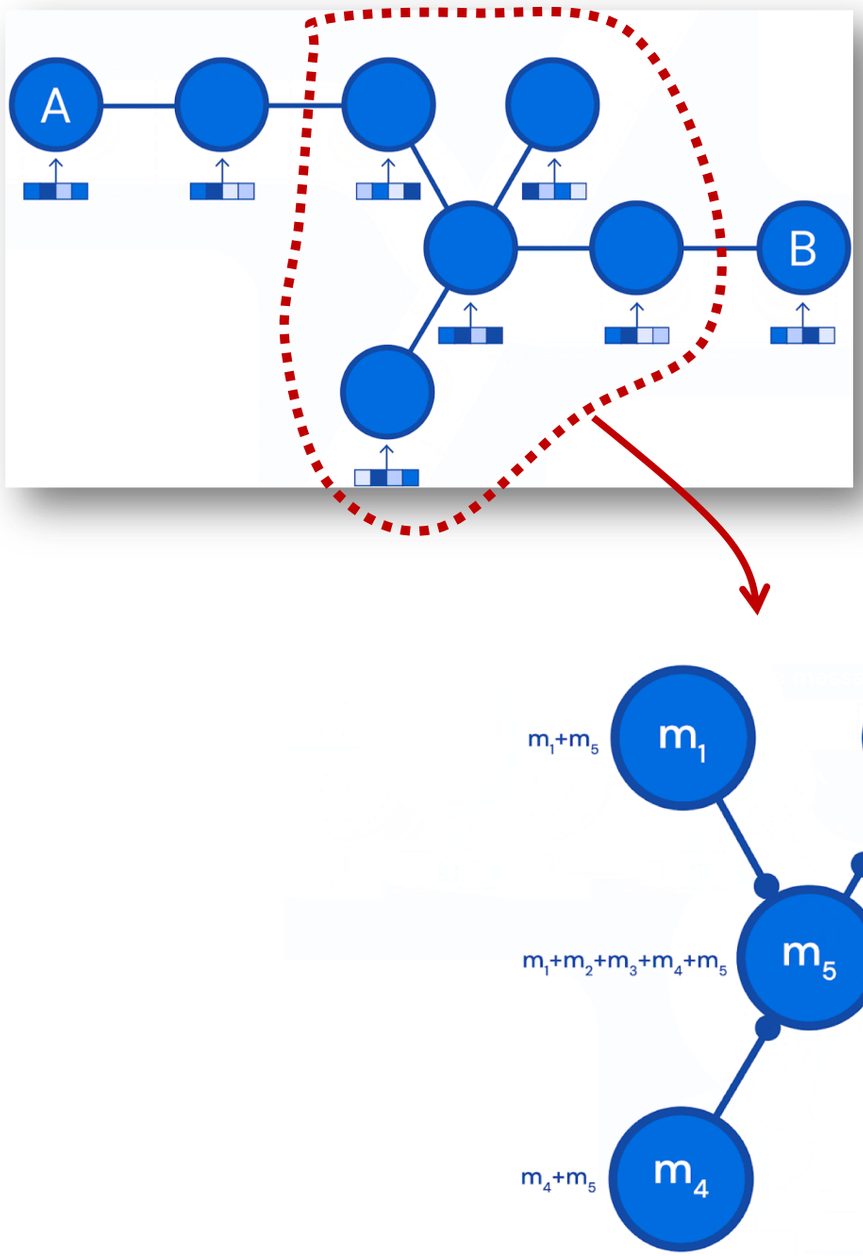
Big picture of GNNs

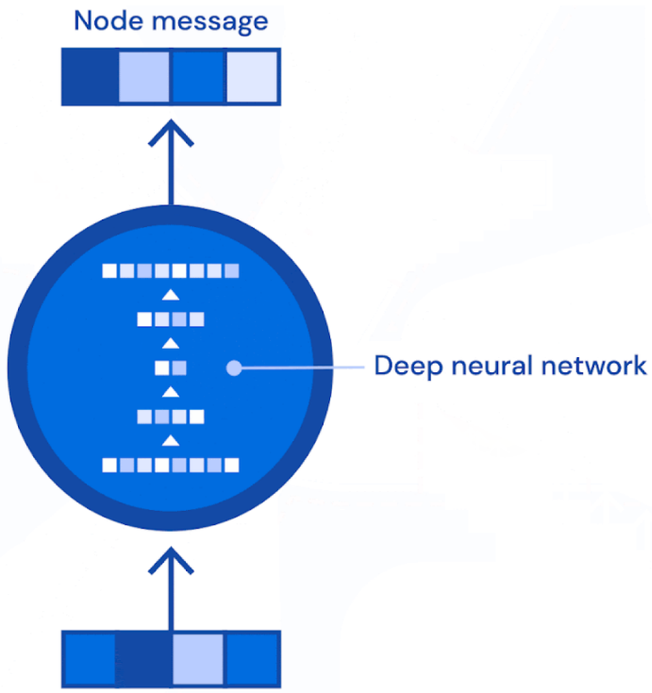


1. Node embeddings:
encode the local neighborhood of each node separately via aggregations

2. Readout:
encode the entire graph by aggregating node embeddings

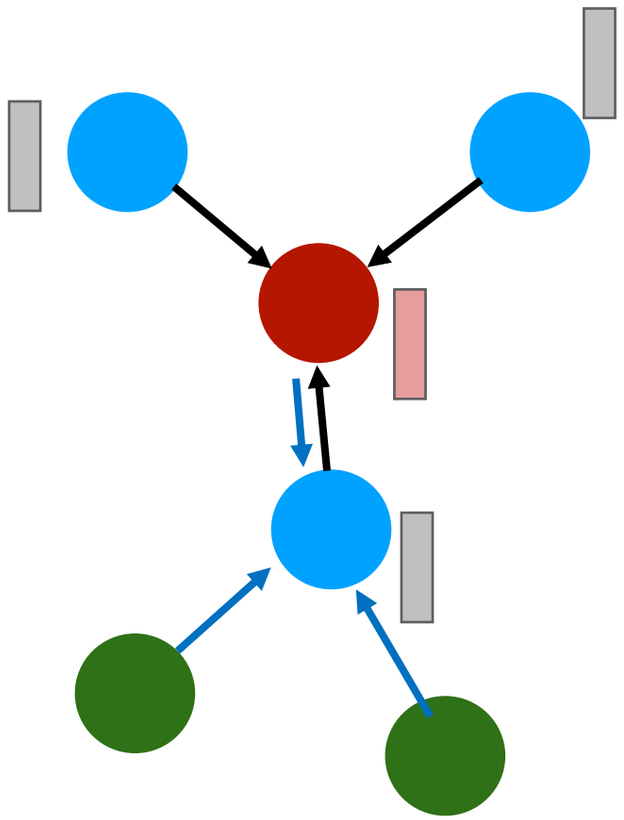
Encoding neighborhoods: message passing





https://deepmind.com/blog/article/traffic-prediction-with-advanced-graph-neural-networks

Encoding neighborhoods: general form



In each round *k*:

Aggregate over neighbors

feature description of node u in round k-1

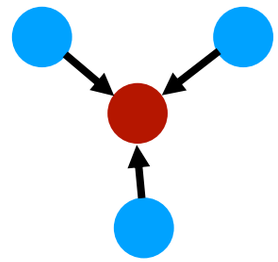
$$m_{\mathcal{N}(v)}^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right)$$

Update: Combine with current node

$$h_v^{(k)} = \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, m_{\mathcal{N}(v)}^{(k)}\right)$$

(Battaglia et al., 2016; Defferrard et al., 2016; Duvenaud et al., 2015; Hamilton et al., 2017a; Kearnes et al., 2016; Kipf & Welling, 2017; Li et al., 2016; Velickovic et al., 2018; Verma & Zhang, 2018; Ying et al., 2018b; Zhang et al., 2018)

Examples of aggregations

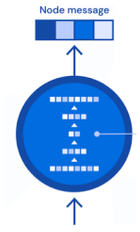


- Sum, average:

$$m_{\mathcal{N}(v)} = \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} h_u$$

Input node attribute of neighbors

- Coordinate-wise min/max
- General form (with small neural networks = MLPs):



$$m_{\mathcal{N}(v)} = \mathrm{MLP}_\theta \left(\sum_{u \in \mathcal{N}(v)} \mathrm{MLP}_\phi(h_u) \right)$$

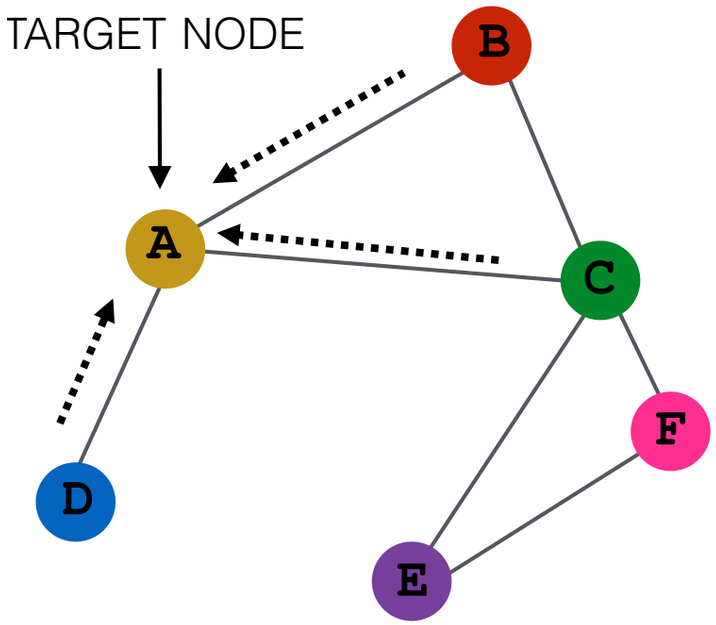
Learned aggregation function

- Update (combine):

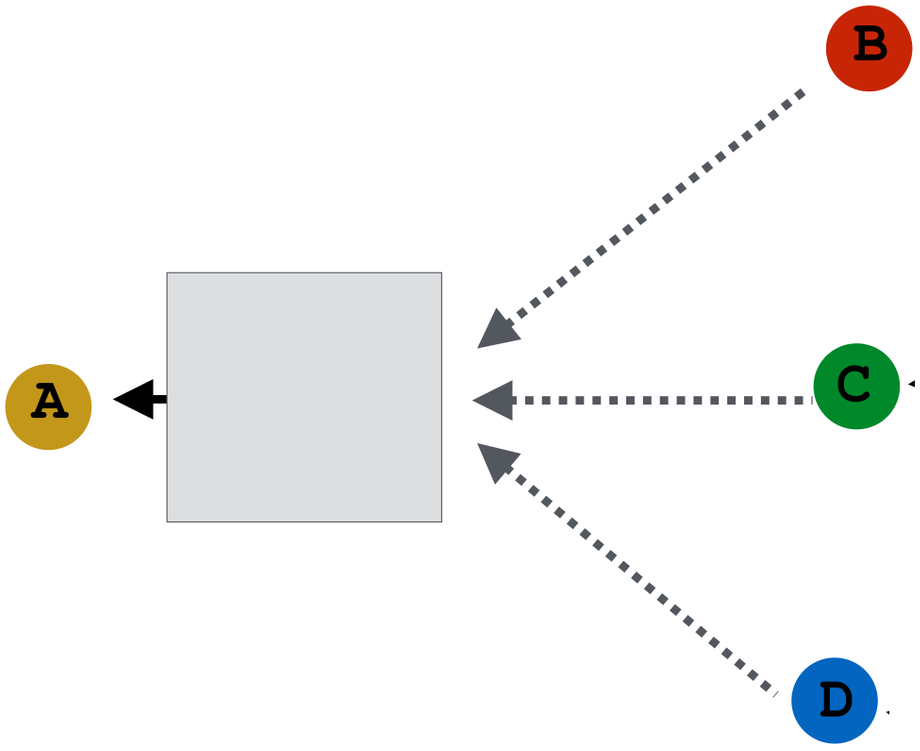
$$h_v^{(k)} = \sigma \left(W_{\text{self}} h_u^{(k-1)} + W_{\text{neigh}} m_{\mathcal{N}(v)}^{(k)} + b \right)$$

learned

Node embeddings as neural network



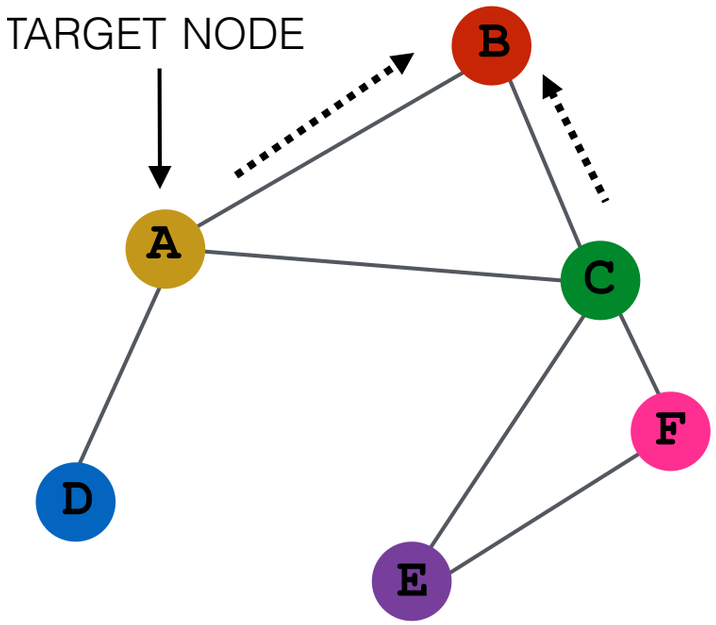
INPUT GRAPH

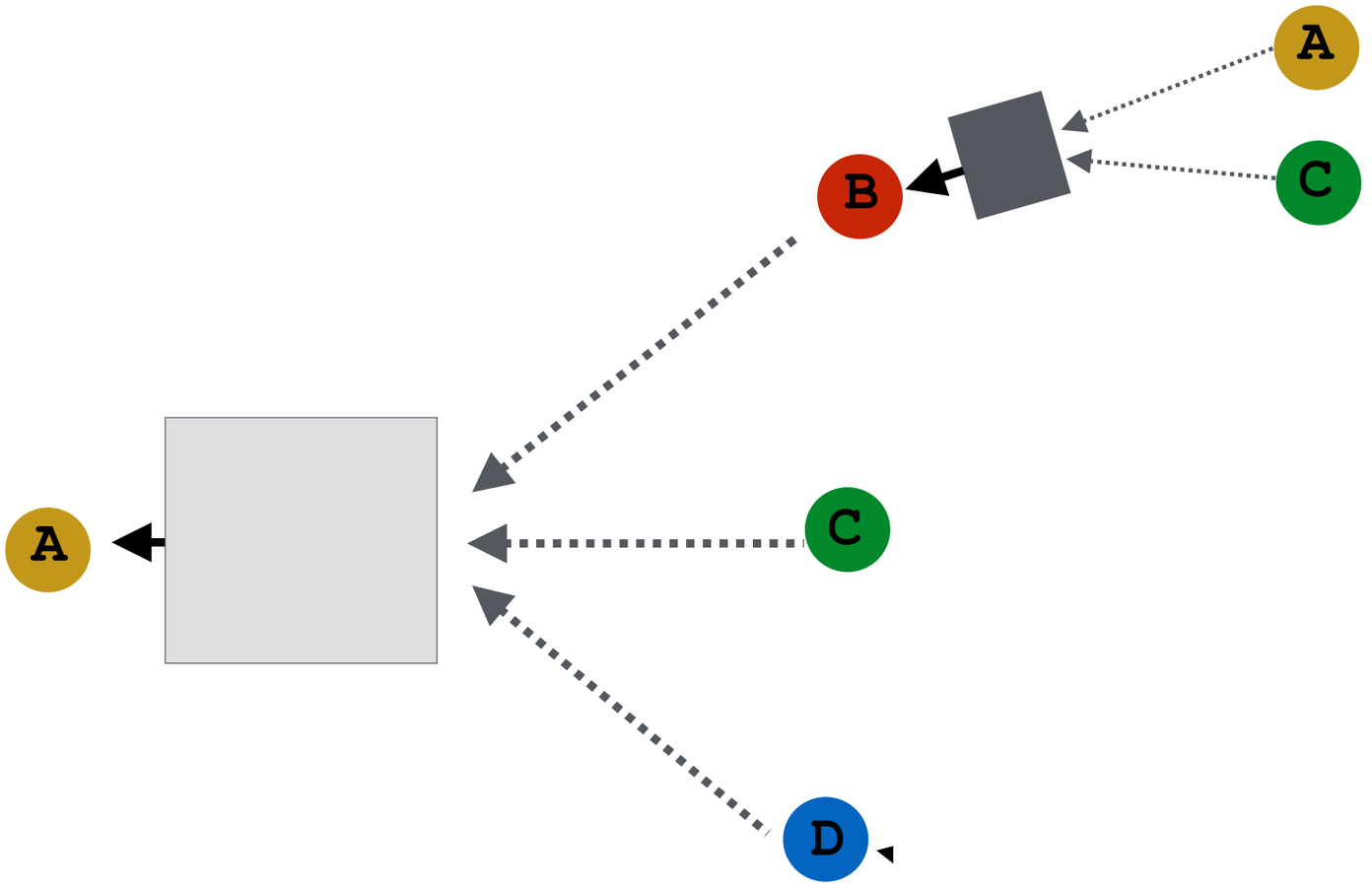


grey boxes: aggregation functions that we learn

(illustrations: J. Leskovec)

Node embeddings as neural network

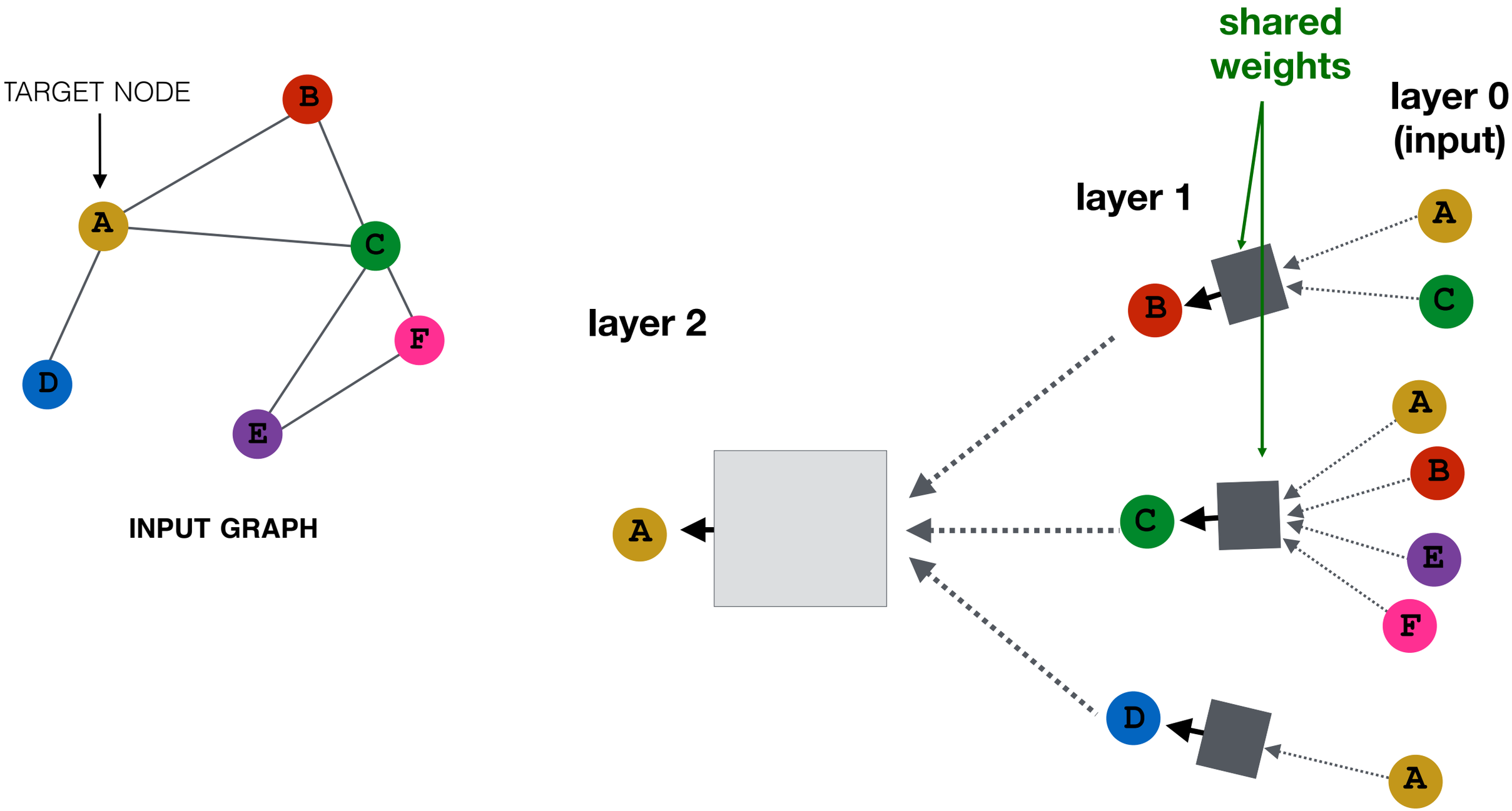




grey boxes: aggregation functions that we learn

(illustrations: J. Leskovec)

Node embeddings as neural network

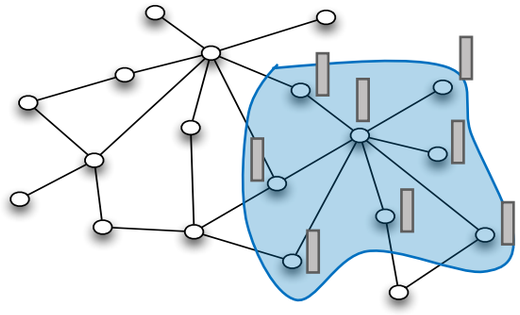


grey boxes: aggregation functions that we learn

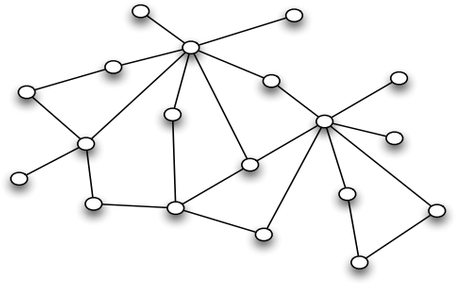
(illustrations: J. Leskovec)

Training a GNN

- **What is a data point?**



Node + its neighborhood (computation tree)



Full graph

- **What to specify?**
  - Aggregate, combine and readout functions
  - Loss function on prediction

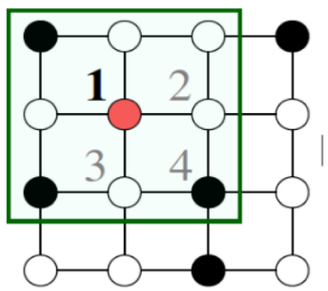
- **Train** with SGD

Summary: GNN model

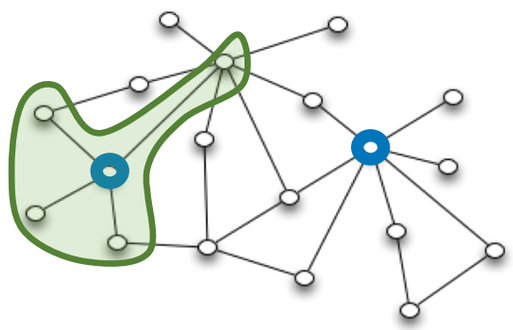
- For many graph prediction problems, information about local neighborhoods adds important information
- Encode local neighborhoods via a general form of “averaging”: aggregation
- We learn the aggregation function via Stochastic Gradient Descent

Comparison to CNN:

- Image = a graph where each node has the same number of neighbors
- Both encode local neighborhoods
- CNN does weighted average, GNN treats each neighbor the same



vs.



Implementing GNNs

- **PyTorch Geometric**:
  https://github.com/rusty1s/pytorch_geometric

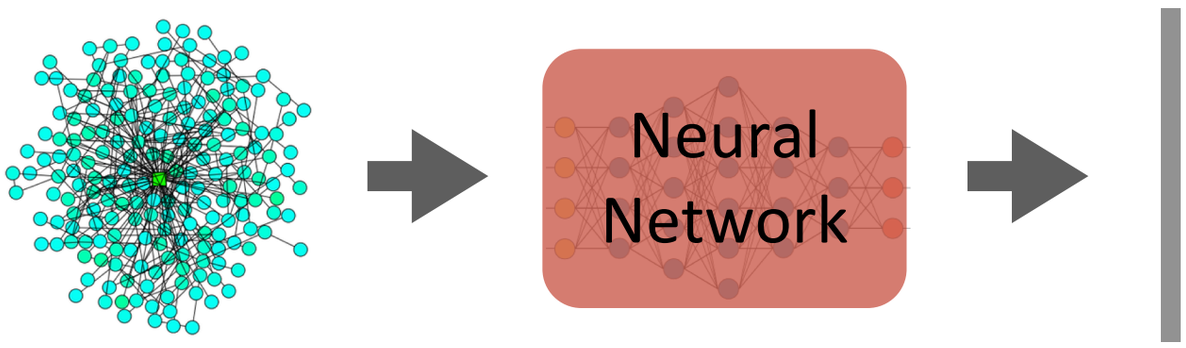
  Introductory code example:
  https://pytorch-geometric.readthedocs.io/en/latest/notes/introduction.html

- **Deep Graph library:**
  https://www.dgl.ai

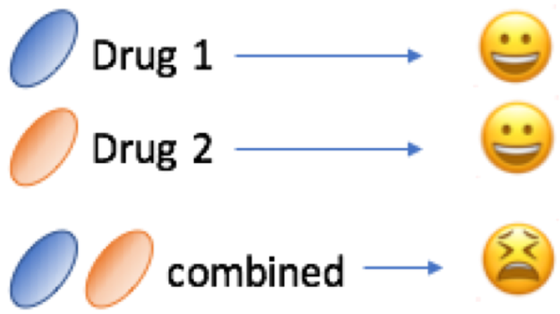
  Introductory code example:
  https://docs.dgl.ai/tutorials/basics/1_first.html

Outline

- Learning tasks with graphs: examples
- Graph neural networks
- Example application:
predicting polypharmacy side effects



Example: Polypharmacy



- high risks of side effects
- 15% of US population affected, annual cost > $177 billion
- difficult to identify:
  rare occurrence, not observed in clinical testing
- Given a pair of drugs, can we predict what types of side effects can occur (if any)?

Modeling Polypharmacy Side Effects with Graph Convolutional Networks

Marinka Zitnik [1], Monica Agrawal [1] and Jure Leskovec [1,2,*]

What information could we use?

- Data on drug-drug interactions (types of side effects)
- Side effects of individual drugs

Side effects are rare. What else can we use?

- Input: feature vectors of 2 drugs
- Output: probabilities for different types of side effects

What information could we use?

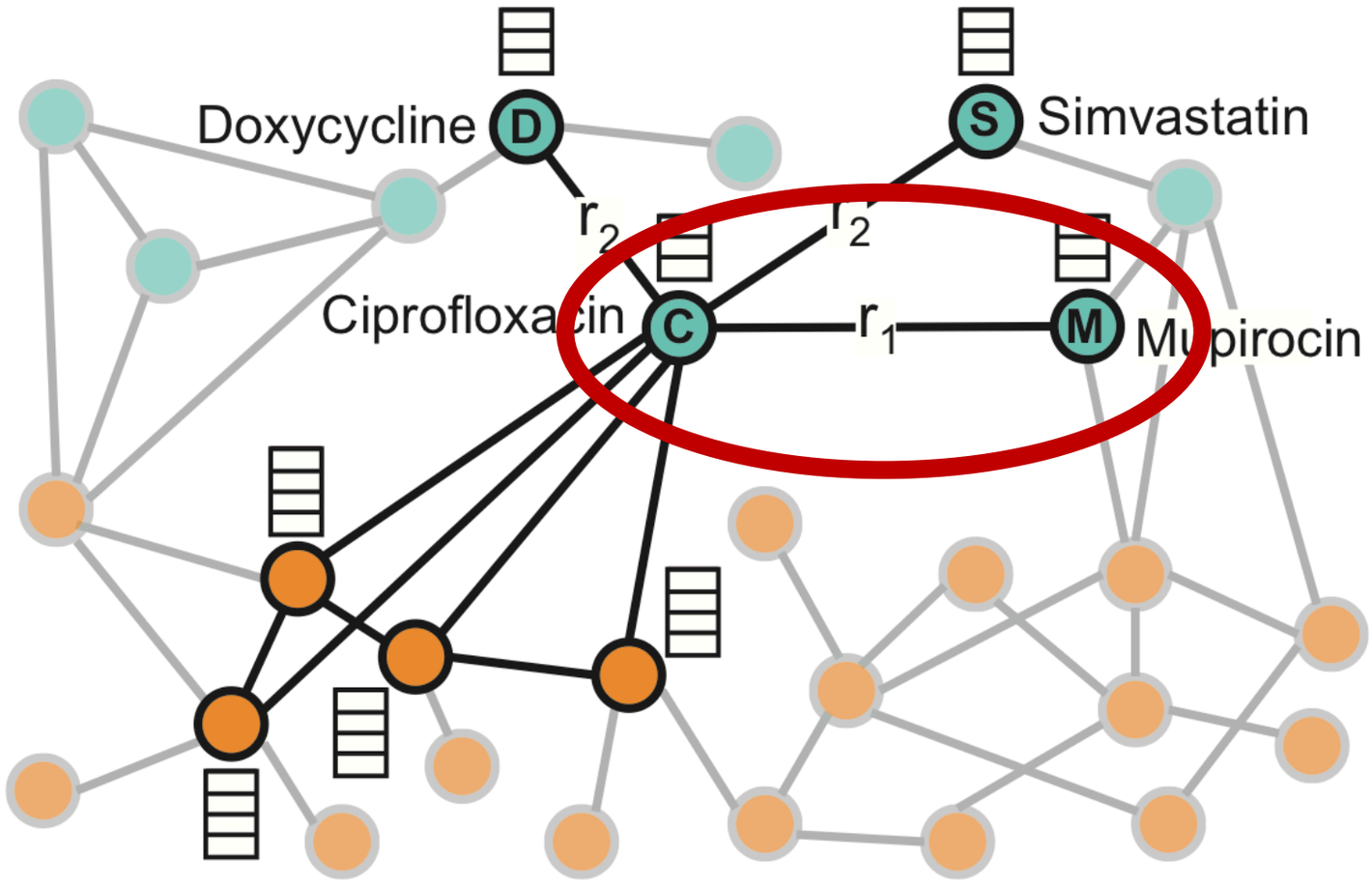
Exploratory analysis reveals:

- Co-prescribed drugs tend to have more target proteins in common than random drug pairs
- In general, >68% of drugs have no target protein in common
  - ➢ *Use drug-drug, drug-protein and protein-protein interactions as information: network*
  - ➢ *Use multiple data sources*

- Wide range of how frequently certain side effects occur in combinations (>53% of side effects in <3% of combinations). Little data on rarer side effects
- Side effects do not occur independently of each other (e.g. hyper-tension co-occurs with anxiety, but negatively correlated with fever)

  - ➢ *Model and predict side effects jointly*

Data



Task: predict edge label

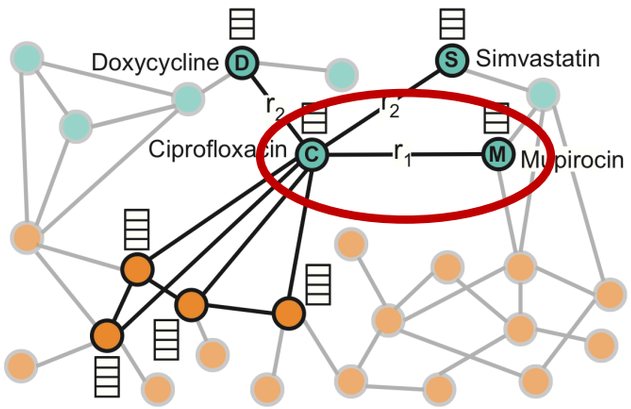
Drugs

Proteins

- 964 types of polypharmacy side effects
- 645 drug nodes, 19,085 protein nodes
- edges: 715,612 prot-prot, 4,651,131 drug-drug, 18,596 drug-protein

Goal: *we want an encoding of (drug) nodes -- use a Graph Neural Network*

Model: overview



Encoder: Outputs embedding for each node. *Graph Neural Network*

Decoder: Takes two node embeddings and predicts probability of each side effect, jointly. *tensor factorization model*

Predictions

$p(C, r_1, S)$

$p(C, r_2, S)$

$p(C, r_3, S)$

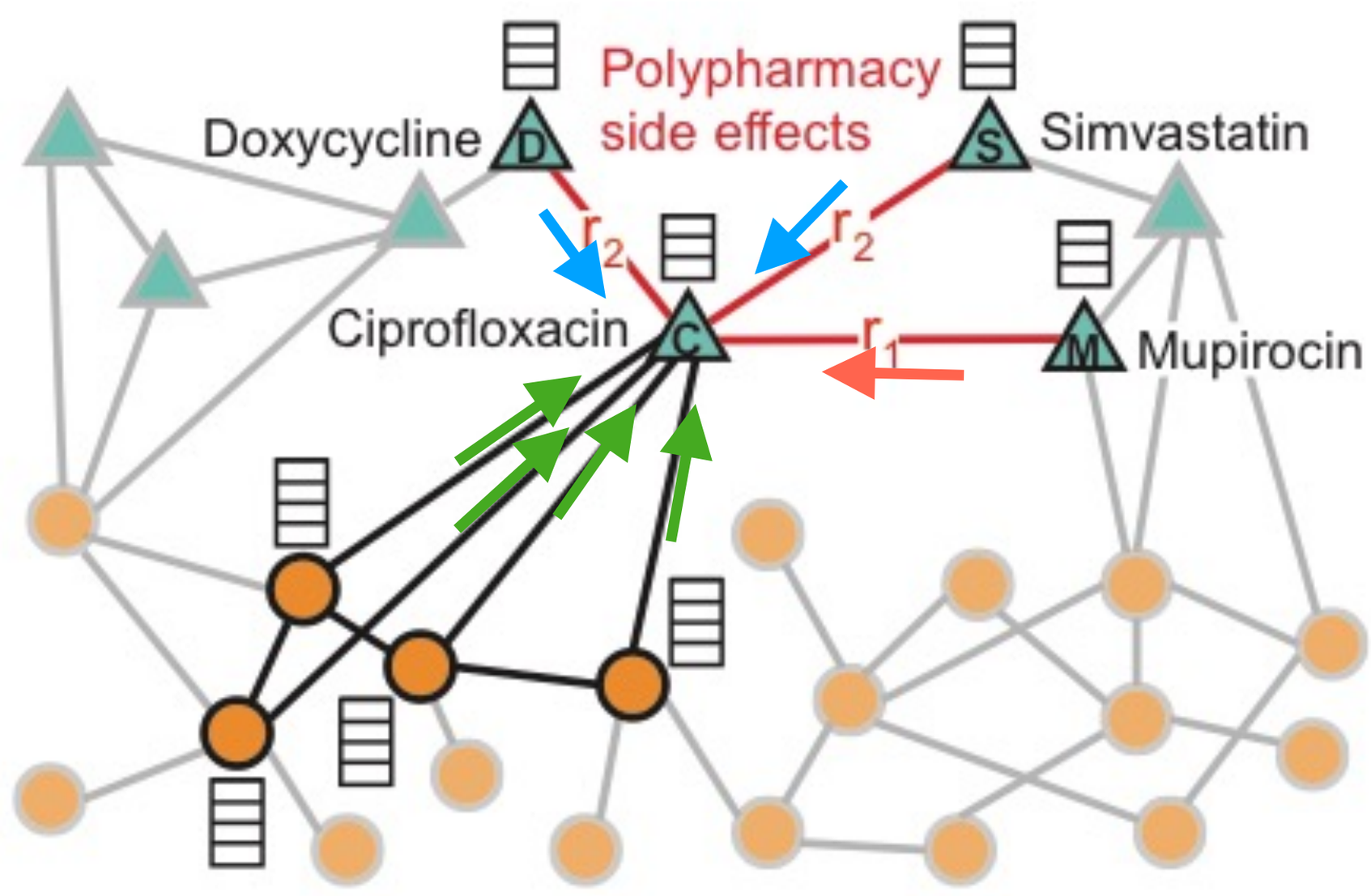
$p(C, r_4, S)$

$p(C, r_n, S)$

Full model trained “end to end”
(= all at once)

Recommender system (such models will be discussed by Prof Shah)

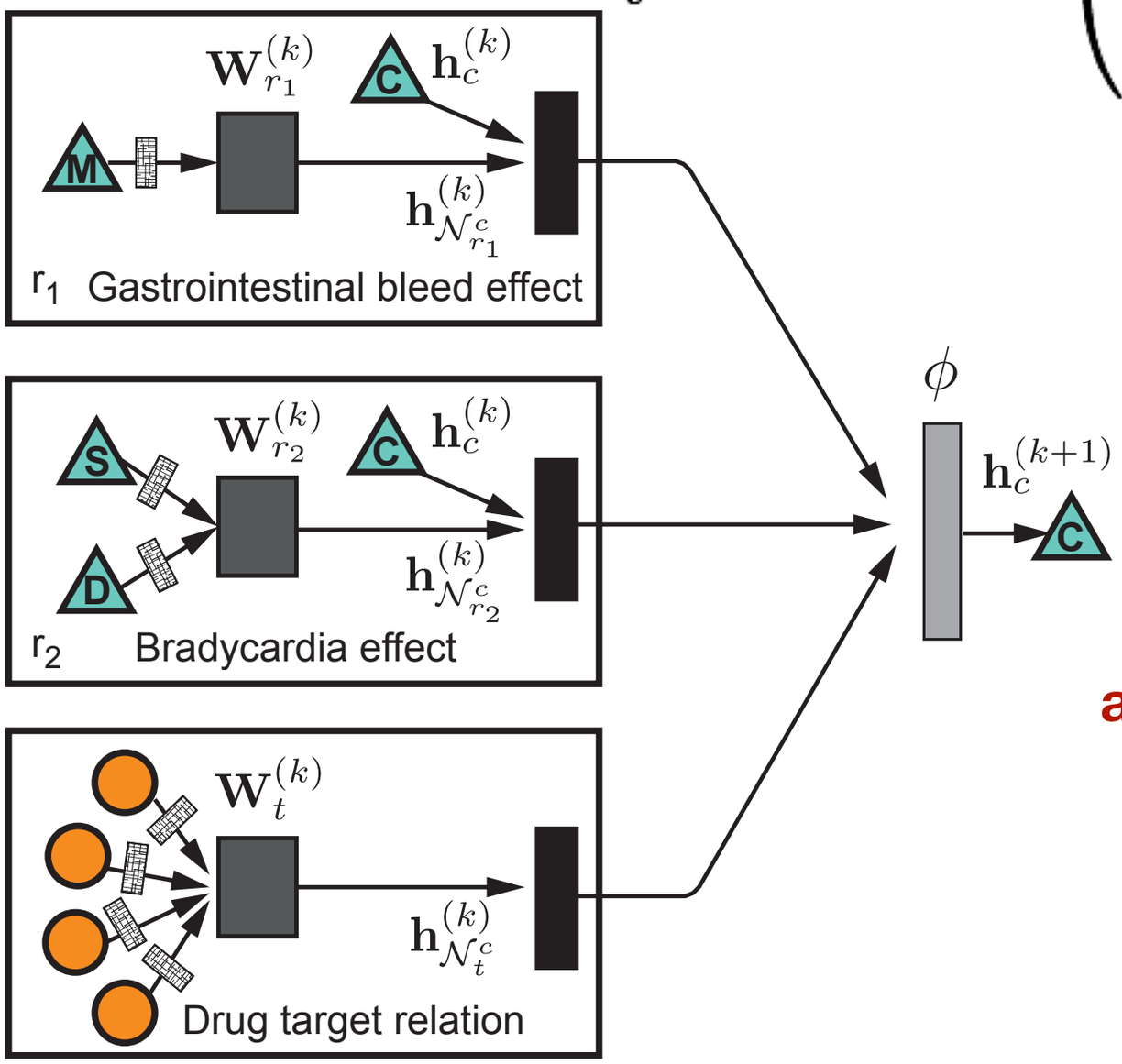
How do we aggregate from neighbors?



- Different types of edges: one for each type of side effect, one for drug-protein
- Aggregate for each edge type separately

How do we aggregate from neighbors?

$$h_v^{(k+1)} = \text{ReLu}\left(\sum_r \sum_{u \in \mathcal{N}_r(v)} c^{vu} W_r^{(k)} h_u^{(k)} + c_r^v h_v^{(k)}\right)$$



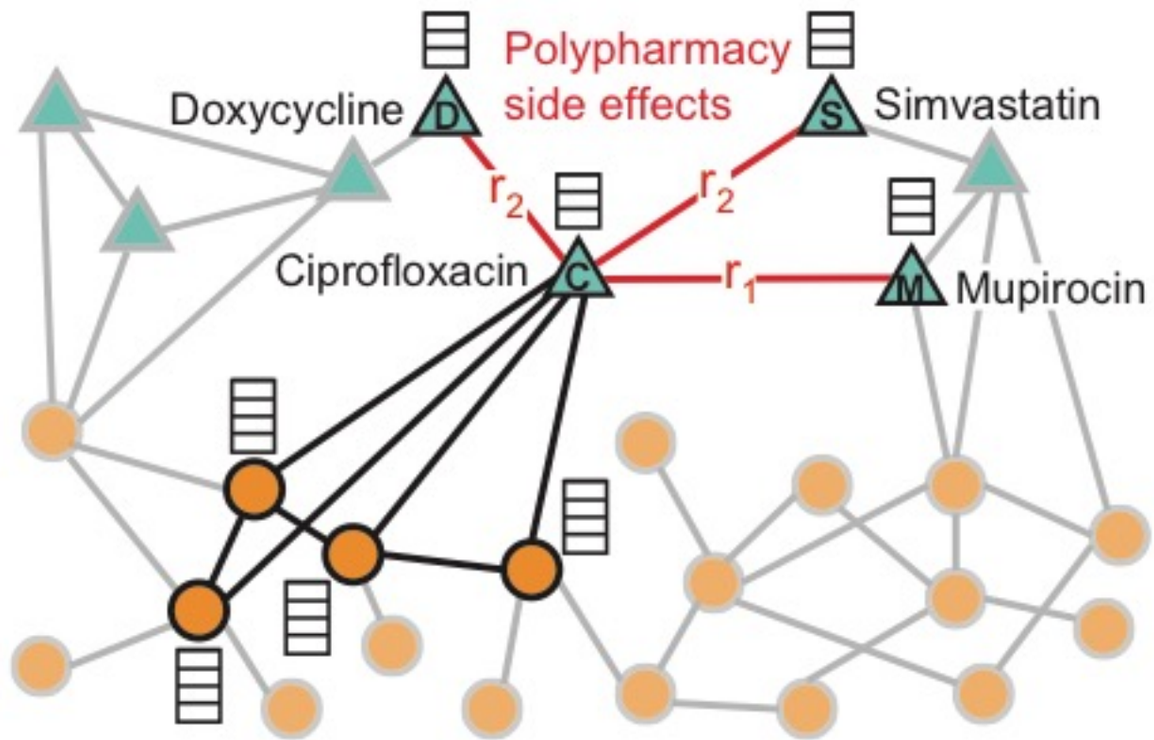
transform embedding of neighbor u in layer k
W is a learned set of parameters

embedding of node v in layer k

aggregate separately for each edge type, then sum it all up and apply ReLu

Training

- What are training data points here?
- *edge queries (u,r,v)*
  (node pair u,v and relation r)
- Label imbalance: side effects are rare (more “no” than “yes”)
- *“negative sampling”*
  (subsample no’s)
- *Optimization algorithm:*
  Adam



Evaluation

1. **Prediction performance**: much better than state of the art
2. **Which are the best/worst predictions?**
   Performs especially well for side effects with molecular underpinnings
3. **literature-based evaluation** of new predicted edges

k	Polypharmacy effect r	Drug i	Drug j	Evidence
1	Sarcoma	Pyrimethamine	Aliskiren	Stage *et al.*
4	Breast disorder	Tolcapone	Pyrimethamine	Bicker *et al.*
6	Renal tubular acidosis	Omeprazole	Amoxicillin	Russo *et al.*
8	Muscle inflammation	Atorvastatin	Amlodipine	Banakh *et al.*
9	Breast inflammation	Aliskiren	Tioconazole	Parving *et al.*

Summary

- Networks/graphs: ubiquitous data type with many applications
- **Graph NNs**: find a vector encoding (embedding) for nodes and the entire graph
- Core: learned aggregation operation over neighbors

- Example modeling application: polypharmacy side effects
  - Prior exploratory analysis / prior knowledge can help design better models
  - Accumulating data from multiple sources can be useful

- More reading / toolboxes:
  - https://www.dgl.ai/
  - https://github.com/rusty1s/pytorch_geometric
  - https://github.com/thunlp/GNNPapers