DEEP LEARNING FOR GRAPHS AND SETS

Adriana Romero

March 26th, 2019 @ UPC School

Bachelor → MSc → PhD → post-doc → Research Scientist, Adjunct Professor

Bachelor

MSc

PhD

Model Compression (FitNets)

Unsupervised pre-training (EPLS)

Applications:
- Image classification
- Semantic segmentation
- Remote sensing

post-doc

Biomedical challenges:
- High dimensional data (genomics)
- Structured prediction (imaging)
- Graph-structured data (meshes)

Research Scientist

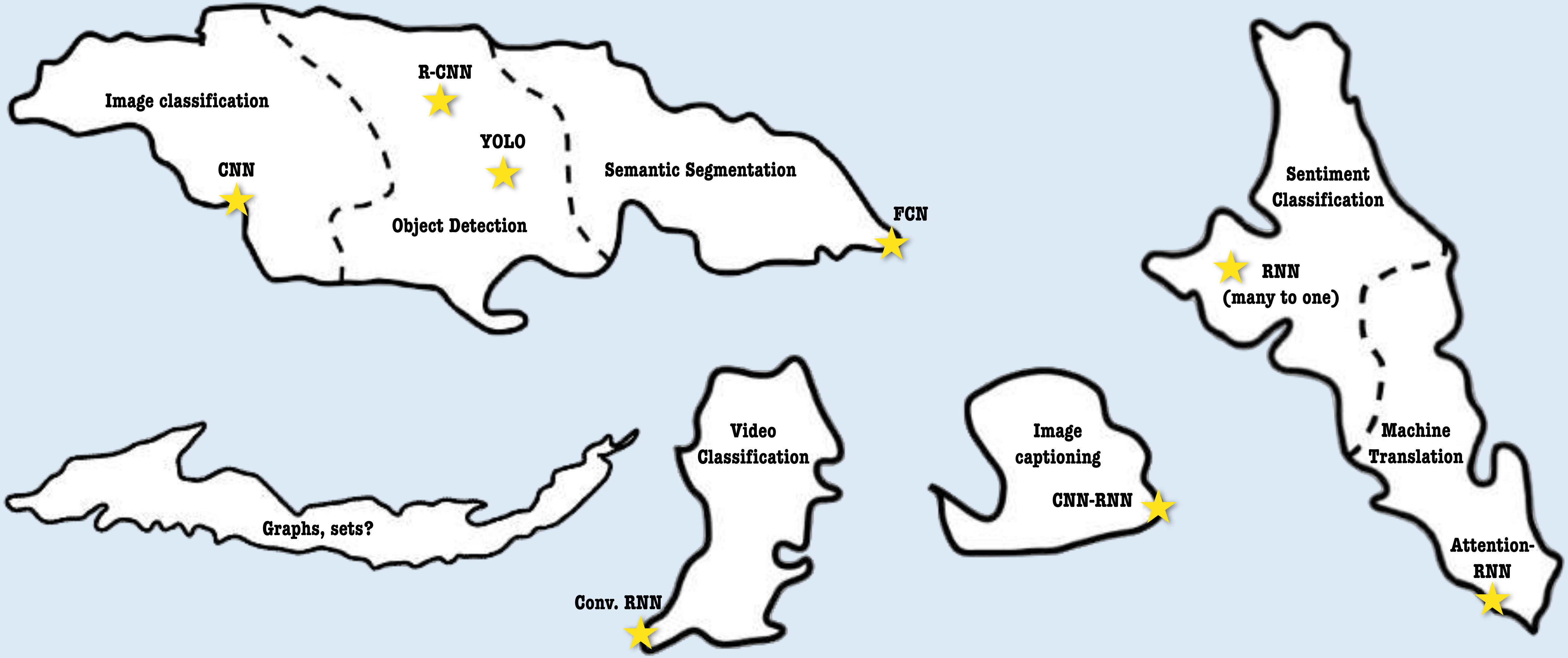
Adjunct Professor

- Conditional generation
- Data multi-modality
- Graphs and sets

PREREQUISITES

- MLP
- CNN
- RNN
- Attention
- How to train neural networks

DEEP LEARNING MODELS & APPLICATIONS



OUTLINE

GRAPH-STRUCTURED DATA:

- Motivation and problem formulation
- Overview of prior work
- Graph attention networks
- Results
- Other graph applications

SET PREDICTION

- Motivation and problem formulation
- Deep learning-based set prediction models, losses and cardinality estimation
- Results

RECIPE GENERATION

- Motivation and problem formulation
- Model
- Results

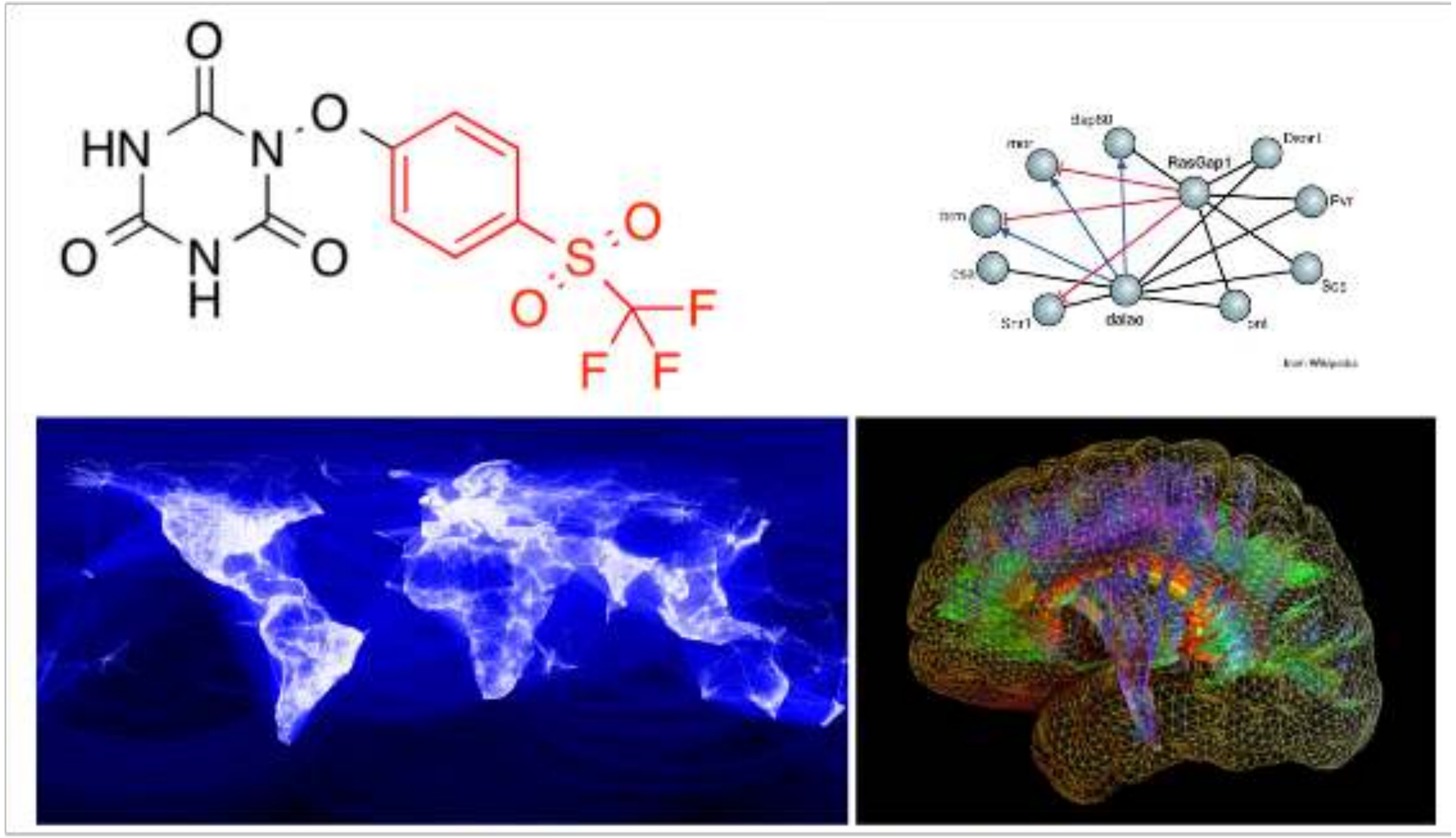
GRAPH-STRUCTURED DATA

P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio

Graph Attention Networks @ ICLR 2018

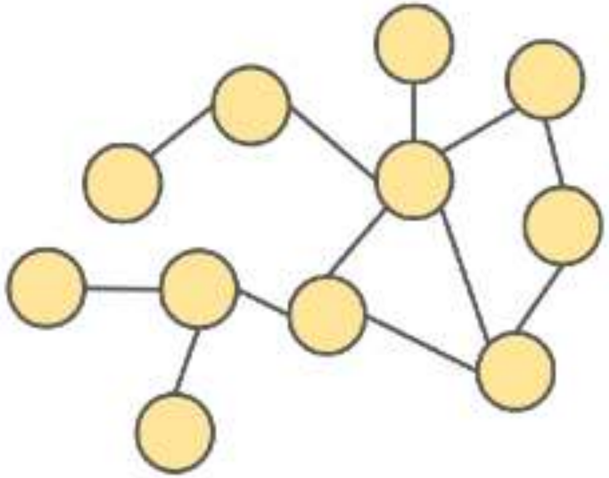
Based on P. Velickovic and G. Cucurull's slides

MOTIVATION



Graphs are everywhere!

PROBLEM FORMULATION



Node classification problem:

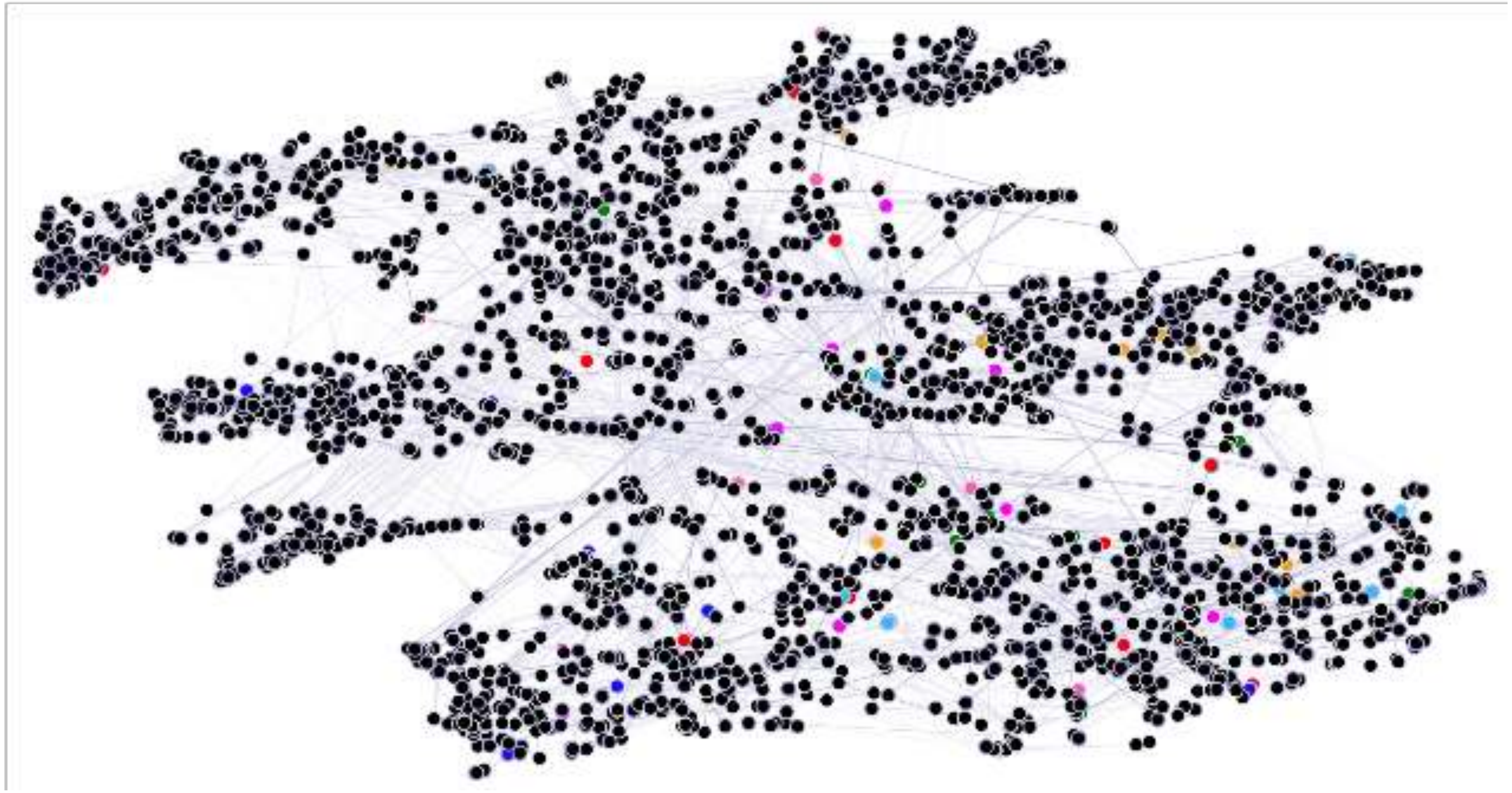
Input: a matrix of node features F and an adjacency matrix A

Output: a matrix of node class probabilities

We assume for simplicity, that edges are unweighted and undirected.

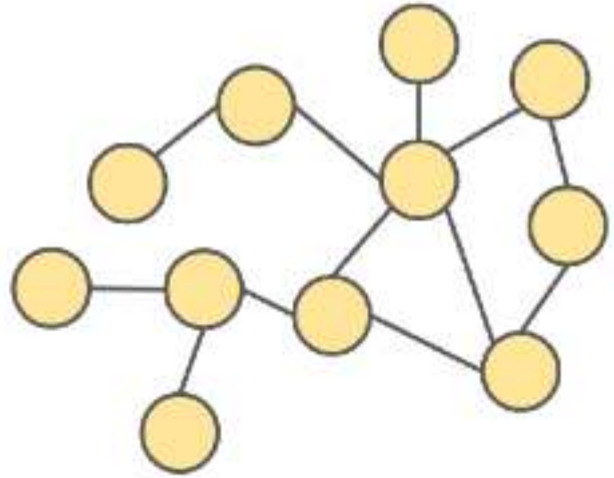
Transductive learning:

PROBLEM FORMULATION



Transductive learning: training sees the features of *all* nodes.

PROBLEM FORMULATION



Node classification problem:

Input: a matrix of node features F and an adjacency matrix A
Output: a matrix of node class probabilities

We assume for simplicity, that edges are unweighted and undirected.

Transductive learning: training sees the features of *all* nodes.

Inductive learning: training does *not* have access to all nodes upfront.

NEURAL NETWORKS FOR GRAPHS

per-node classifier

Does not exploit graph structure!

NEURAL NETWORKS FOR GRAPHS

per-node classifier

per-node classifier +
node similarity constraint
(Weston et al., 2008)

per-node classifier
based on node & structure features
(Perozzi et al., 2014; Tang et al., 2015;
Grover et al., 2016; Yang et al., 2016)

NEURAL NETWORKS FOR GRAPHS

per-node classifier

per-node classifier +
node similarity constraint
(Weston et al., 2008)

Graph structure injected indirectly!

per-node classifier
based on node & structural features
(Perozzi et al., 2014; Tang et al., 2015;
Grover et al., 2016; Yang et al., 2016)

A good representation of a node should allow
us to easily predict the nodes that *surround* it.

NEURAL NETWORKS FOR GRAPHS

per-node classifier

per-node classifier +
node similarity constraint
(Weston et al., 2008)

per-node classifier
based on node & structural features
(Perozzi et al., 2014; Tang et al., 2015;
Grover et al., 2016; Yang et al., 2016)

A good representation of a node should allow
us to easily predict the nodes that *surround* it.

Graph Neural Networks
(Gori et al., 2005; Scarselli et al., 2009)

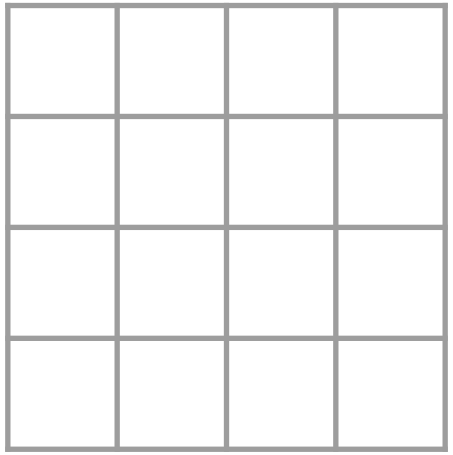
Gated Graph Neural Networks
(Li et al., 2016)

Our graphs have static features!

HOW ABOUT LEVERAGING CONVOLUTIONS? (1)

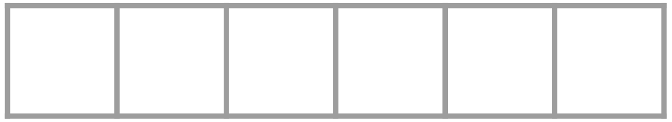
CNN work well on data defined in n-D grids

2D grid



1D grid

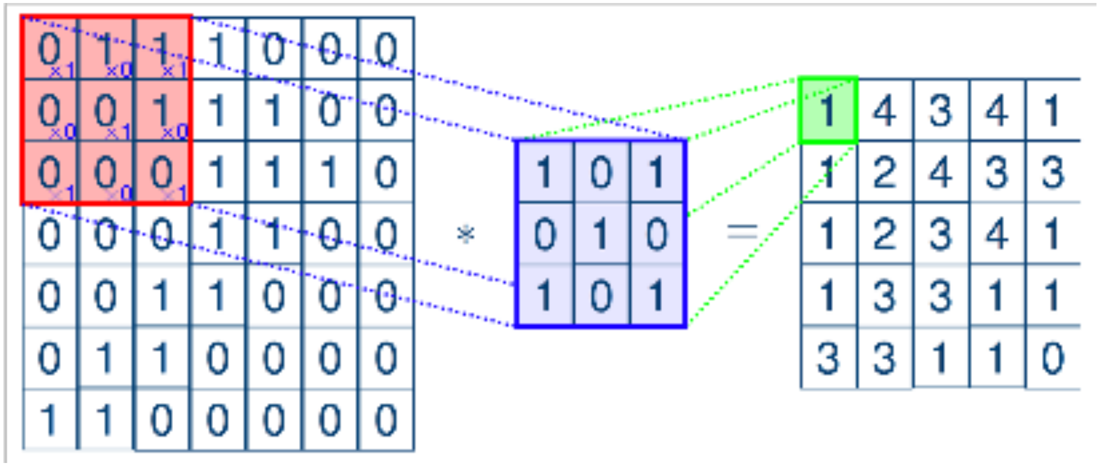
"I like cats and dogs."



HOW ABOUT LEVERAGING CONVOLUTIONS? (2)

Why convolutions?

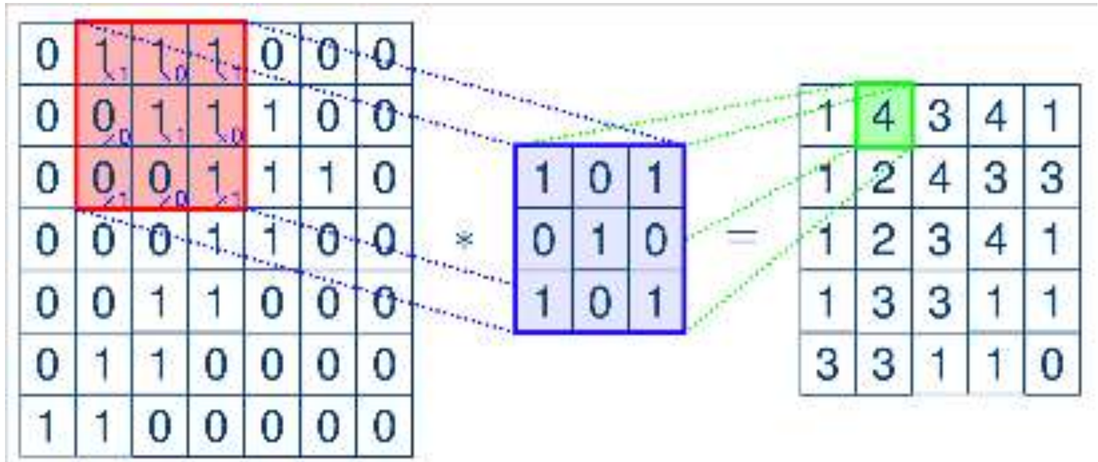
Independent of input size
Fixed # parameters (weight sharing)
Localized (act on local neighborhoods)
Specify different importance per node
Applicable to both transductive and
inductive problems



HOW ABOUT LEVERAGING CONVOLUTIONS? (2)

Why convolutions?

Independent of input size
Fixed # parameters (weight sharing)
Localized (act on local neighborhoods)
Specify different importance per node
Applicable to both transductive and
inductive problems

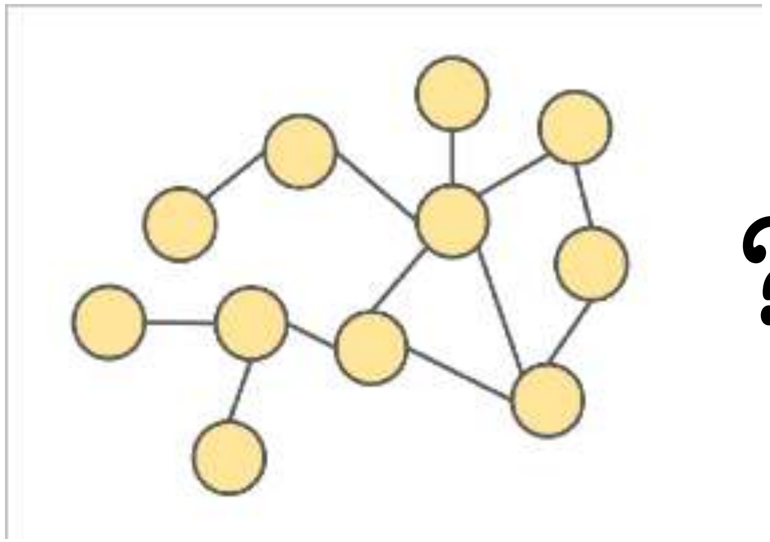


HOW ABOUT LEVERAGING CONVOLUTIONS? (3)

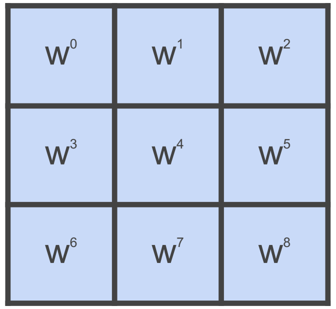
Why convolutions?

Independent of input size
Fixed # parameters (weight sharing)
Localized (act on local neighborhoods)
Specify different importance per node
Applicable to both transductive and
inductive problems

How do we apply a convolutional filter over a graph?

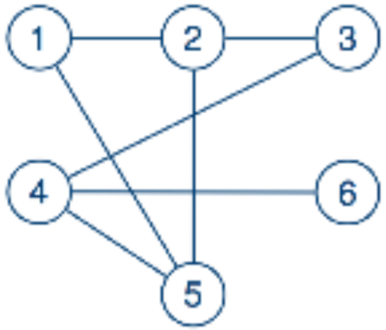


?



GRAPH SPECTRAL NETWORKS

Operating in the graph spectral domain



$$\mathbf{L} = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

- ➢ **Computing U can be expensive.**
- ➢ **Filters are not localized.**

<u>Graph Laplacian</u>:

D (degree matrix) – A (adjacency matrix)

(or a *normalized* version of it)

$$\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^{\mathbf{T}}$$

- Multiplying the feature matrix by $\mathbf{U}^{\mathbf{T}}$ allows us to enter the spectral domain of the graph.
- Convolution amounts to simple multiplication.

$$\vec{h}'_i = \mathbf{U}\left(\vec{w} \odot \mathbf{U}^T \mathbf{W} \vec{h}_i\right)$$

(Bruna et al., 2014)

CHEBYNETS & GCN

Rather than computing the Fourier transform, use Chebyshev polynomials of order k to approximate it.

$$\vec{h}'_i = \sum_{k=0}^{K} w_k T_k(\mathbf{L}) \mathbf{W} \vec{h}_i$$

GCN set to K = 1, further simplifying the definition of a convolutional layer.

- These polynomials have a recursive formulation, highly simplifying the computation.
- T_k is a weighted sum of all powers of L up to L^k. ⟶ K-localized
- The number of parameters is fixed (k).

➢ **Unable to specify different weights to different nodes in a neighborhood.**

(Defferrard et al., 2016; Kipf & Welling, 2017)

NON-SPECTRAL APPROACHES

✓ **Do not assume a particular & fixed graph Laplacian**

Molecular fingerprint networks

- Processing with various degrees by learning a separate weight matrix per node degree.

➢ **Does not scale to graphs with very wide degree distributions.**

GraphSAGE

- Restricts every degree to be the same, by sampling a fixed-size of neighbors.

➢ **Inherently drops relevant data.**

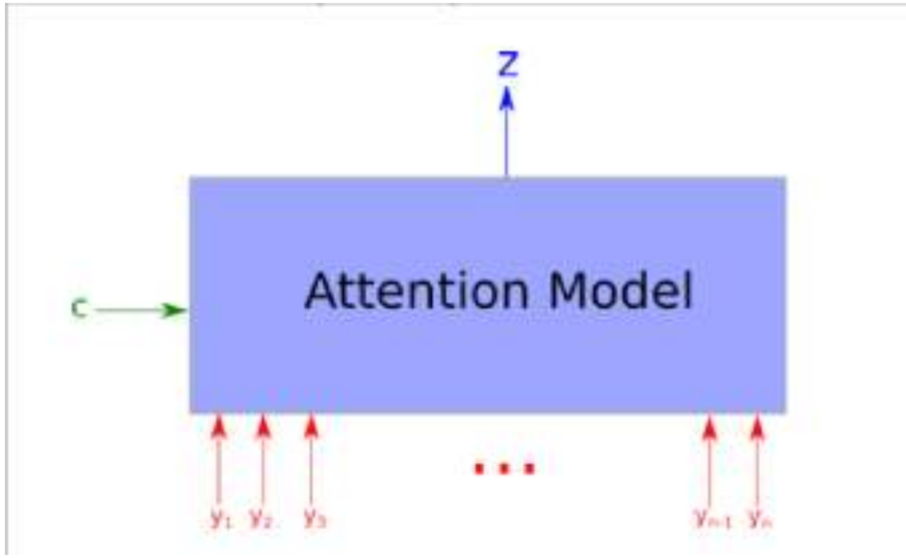
(Duvenaud et al., 2015; Hamilton et al., 2017)

ATTENTION AS A WAY TO OVERCOME THE PREVIOUS ISSUES

ATTENTION (1)

Attention allows us to focus on a subset of information and select the most pertinent piece of it.

Previous hidden state

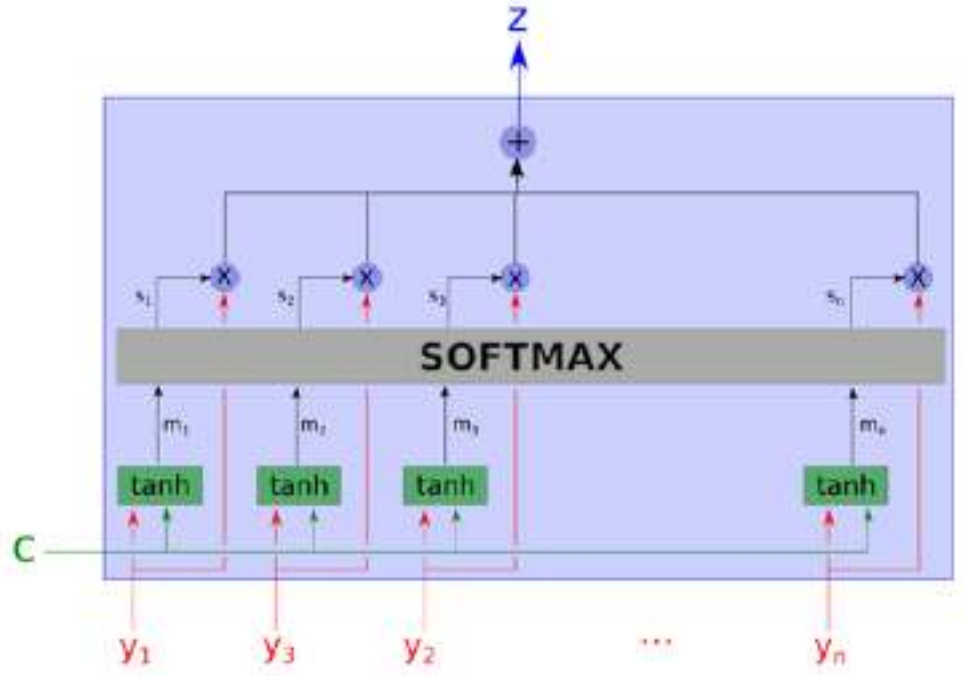


Attention input, can be of variable length
(image features, seq of annotations)

Image from https://blog.heuritech.com/2016/01/20/attention-mechanism/

(Bahdanau et al., 2015; Xu et al. 2015)

ATTENTION (2)



1. Combine input and hidden information

$$\mathbf{m_i} = f(\mathbf{W_c c} + \mathbf{W_y y_i})$$

2. Apply softmax to obtain attention coefficients s_i
3. Apply attention coefficients to input and aggregate

$$\mathbf{z} = \sum_i s_i \odot \mathbf{y_i}$$

Image from https://blog.heuritech.com/2016/01/20/attention-mechanism/

(Bahdanau et al., 2015; Xu et al. 2015)

GRAPH ATTENTION NETWORKS (1)

The idea is to leverage the self-attention operator to *emulate* convolutions on graphs.

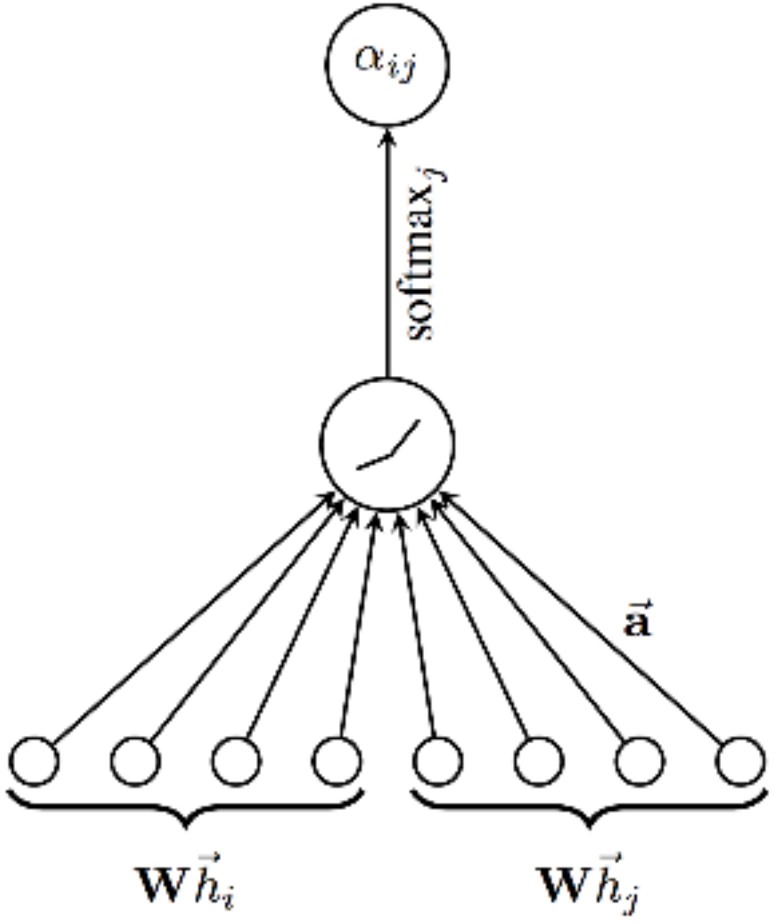
Self attention allows the input to attend over itself:

$$\alpha_{ij} = a(\vec{h}_i, \vec{h}_j)$$
$$\vec{h}'_i = \sum_j softmax_j(\alpha_{ij})\vec{h}_j$$

- A naïve formulation would compute attention coefficients over all pairs of nodes, dropping all structural information.
- We restrict the model to only attend over the node's neighborhood, making the operator localized.

(Velickovic et al., 2018)

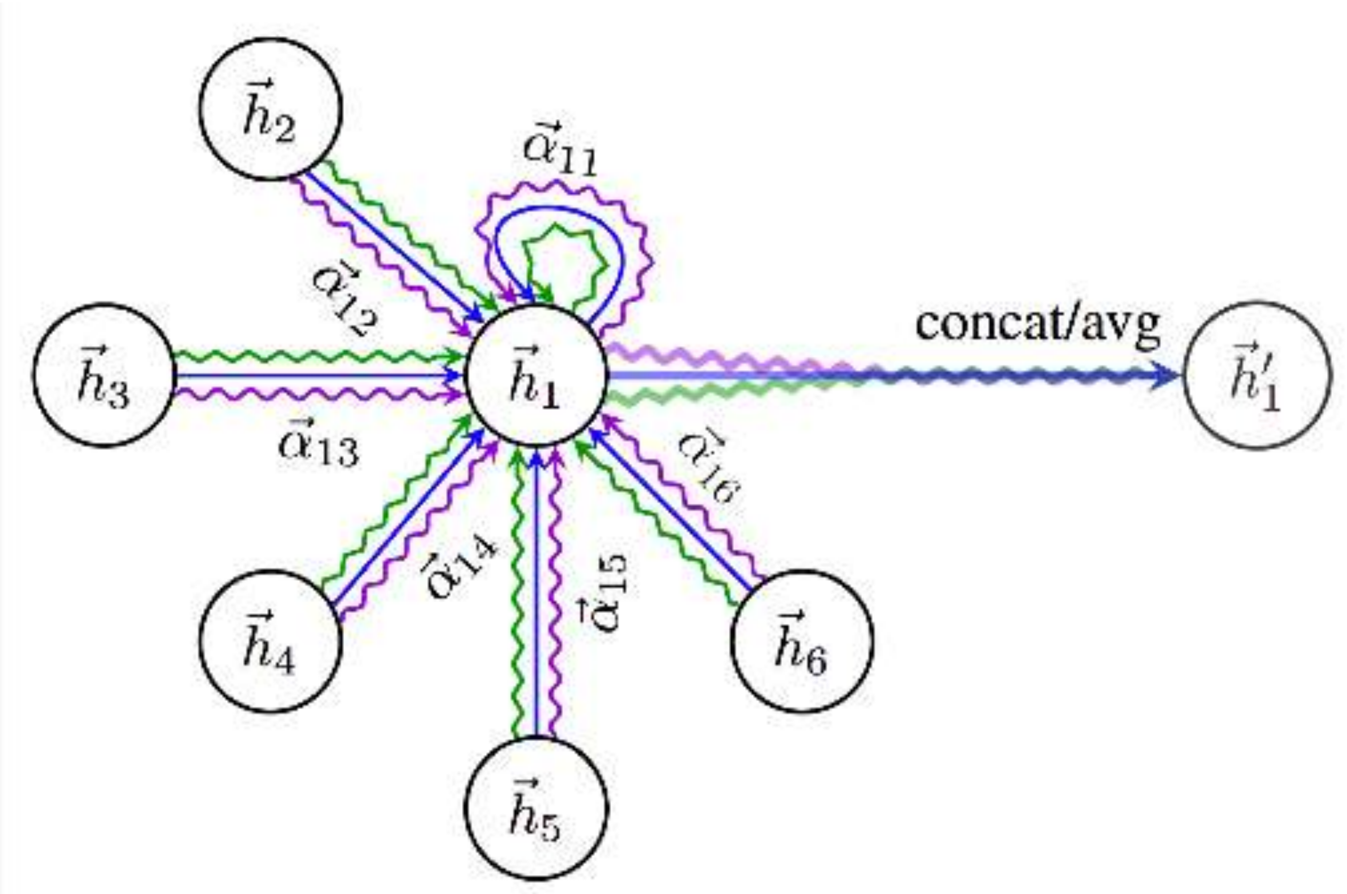
GRAPH ATTENTION NETWORKS (2)



$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$

GRAPH ATTENTION NETWORKS (3)



$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \vec{h}_j\right)$$

For one attention head.

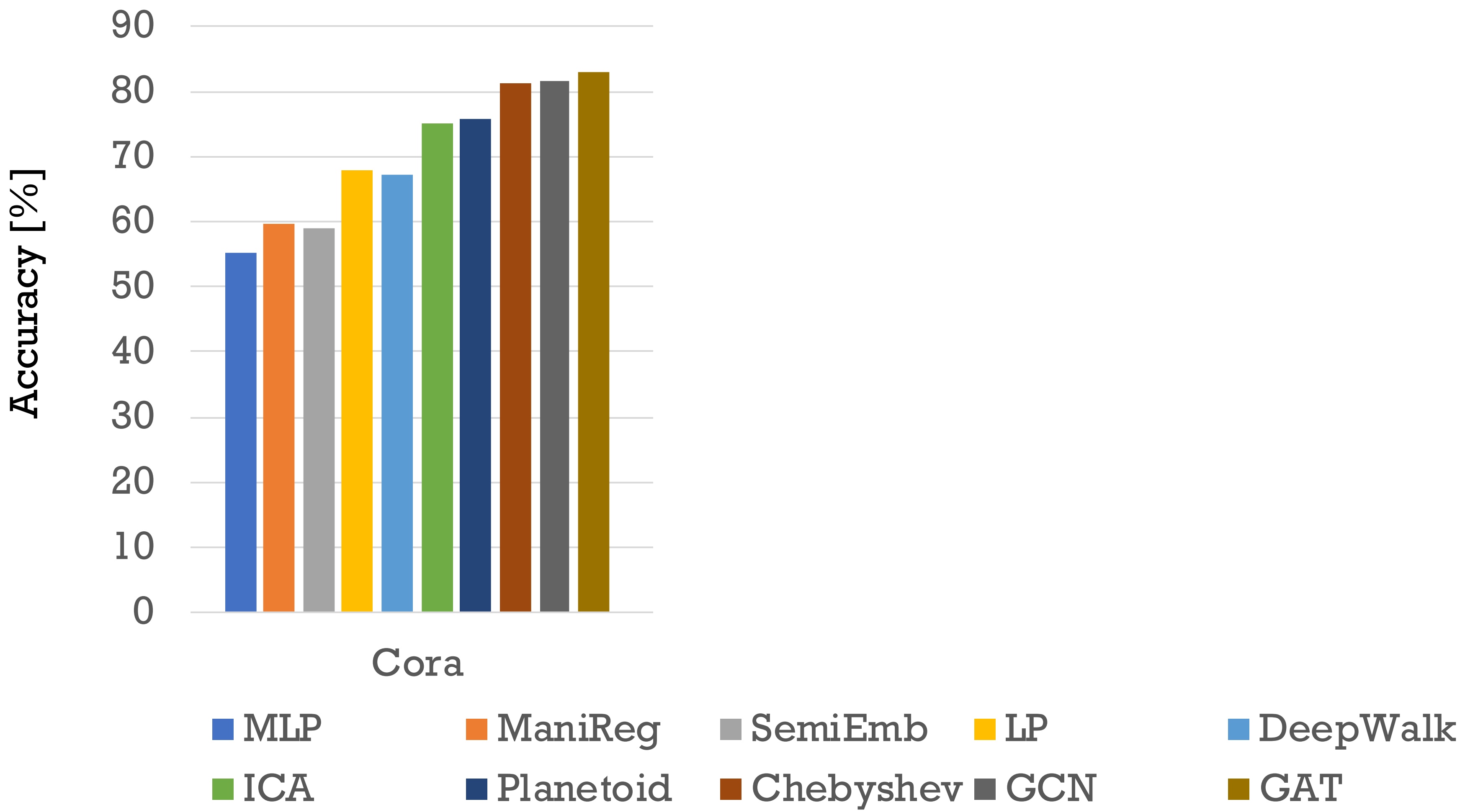
DATASETS (1)

	Cora	Citeseer	Pubmed
# Graphs	1	1	1
# Nodes	2708	3327	19717
# Edges	5429	4732	44338
# Features / Node	1433	3703	500
# Classes	7	6	3
# Training Nodes	140	120	60
# Validation Nodes	500	500	500
# Test Nodes	1000	1000	1000

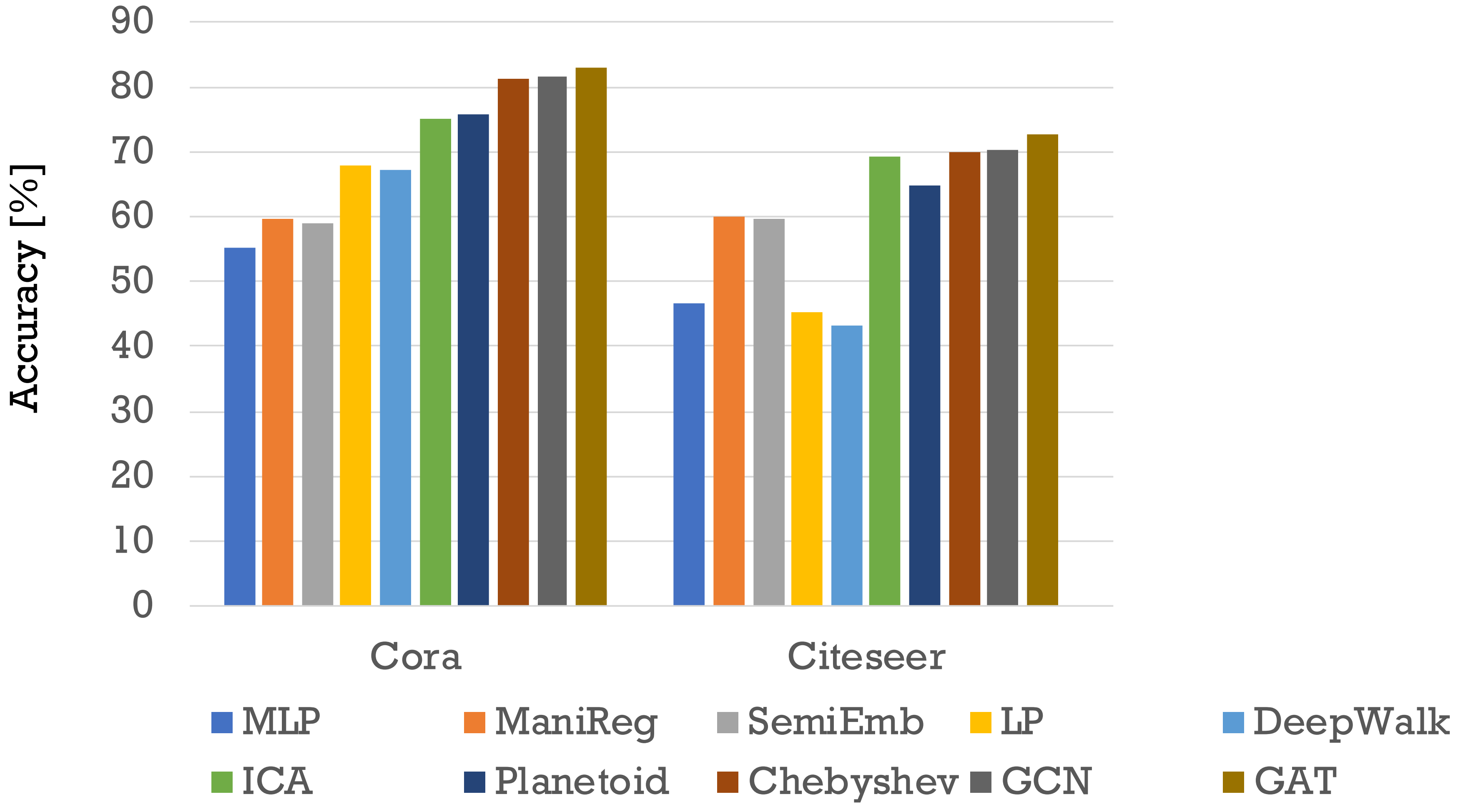
Nodes: documents
Edges: citations

Features: sparse BoW
Output: 1 class per document

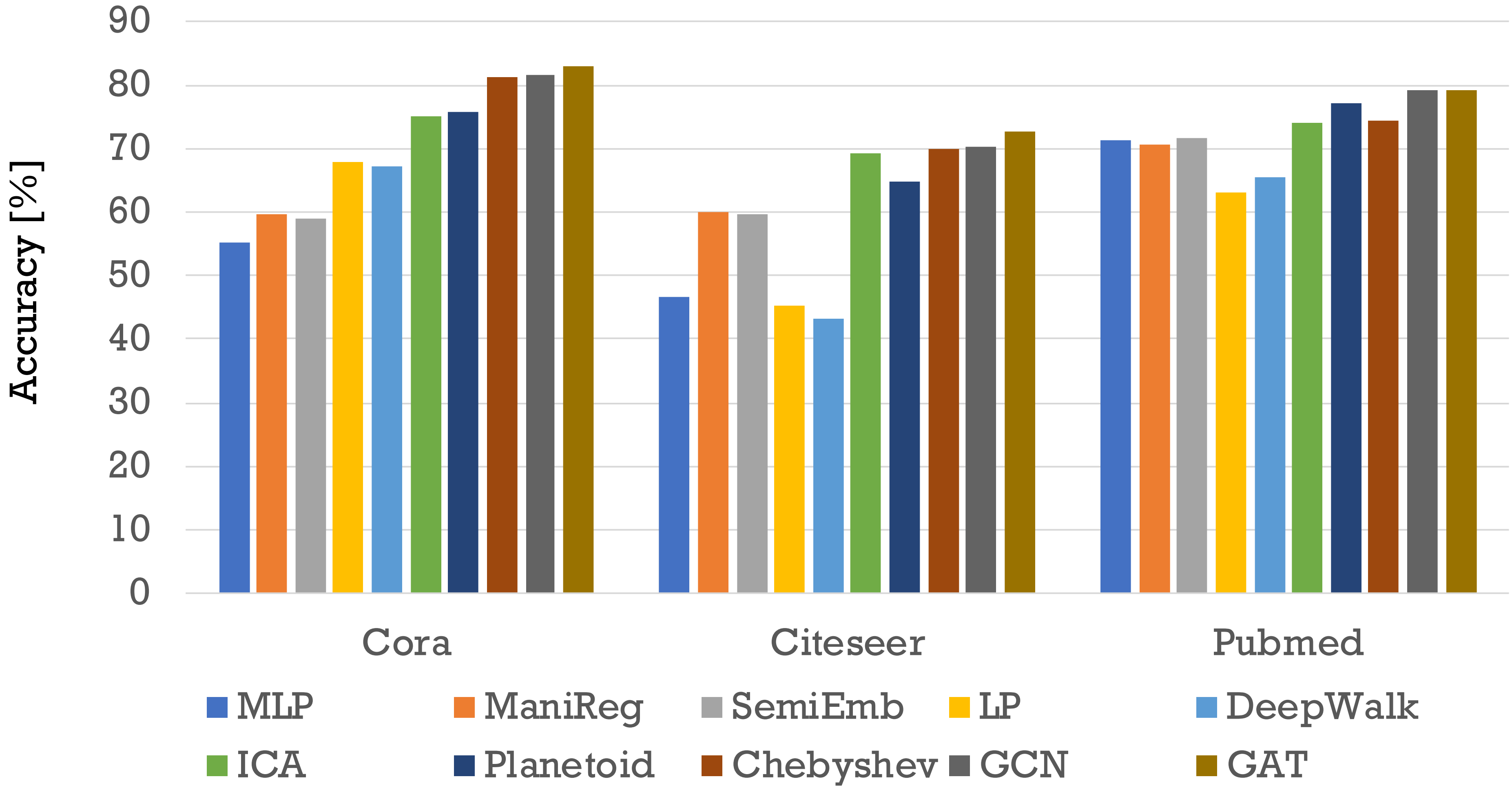
RESULTS (1)


Accuracy [%]
90
80
70
60
50
40
30
20
10
0
Cora
MLP
ManiReg
SemiEmb
LP
DeepWalk
ICA
Planetoid
Chebyshev
GCN
GAT

RESULTS (1)


Accuracy [%]
90
80
70
60
50
40
30
20
10
0
Cora
Citeseer
MLP
ManiReg
SemiEmb
LP
DeepWalk
ICA
Planetoid
Chebyshev
GCN
GAT

RESULTS (1)



90
80
70
60
50
40
30
20
10
0
Accuracy [%]
Cora
Citeseer
Pubmed
MLP
ManiReg
SemiEmb
LP
DeepWalk
ICA
Planetoid
Chebyshev
GCN
GAT

DATASETS (2)

	PPI	HCP
# Graphs	24	100
# Nodes	56944	119500
# Edges	818716	342900
# Features / Node	50	9
# Classes	121 (multilabel)	3
# Training Nodes	44906 (20 graphs)	95600 (80 graphs)
# Validation Nodes	6514 (2 graphs)	11950 (10 graphs)
# Test Nodes	5524 (2 graphs)	11950 (10 graphs)

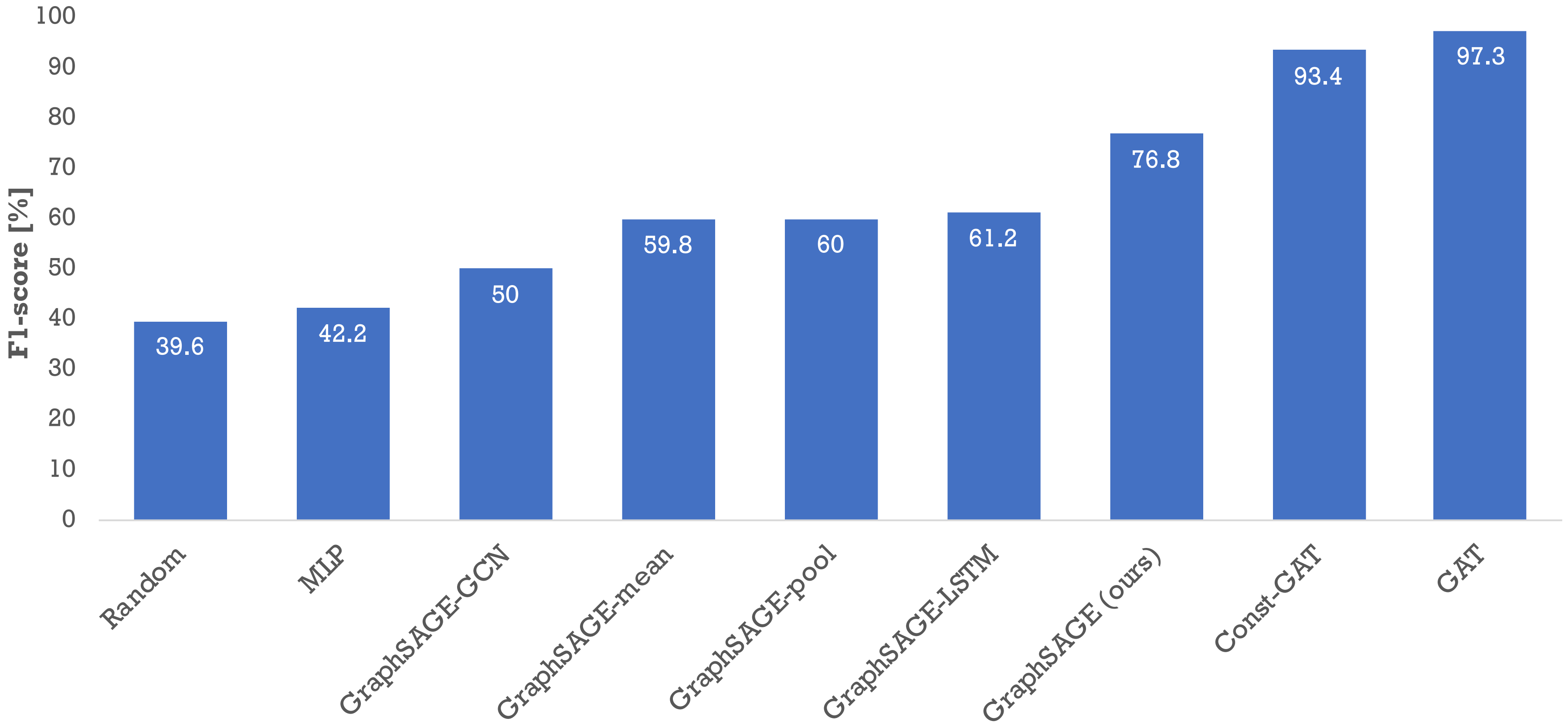
Graphs: one per human tissue
Features: positional gene sets, motif gene sets, immunological signatures
Output: n active labels (protein roles)

DATASETS (2)

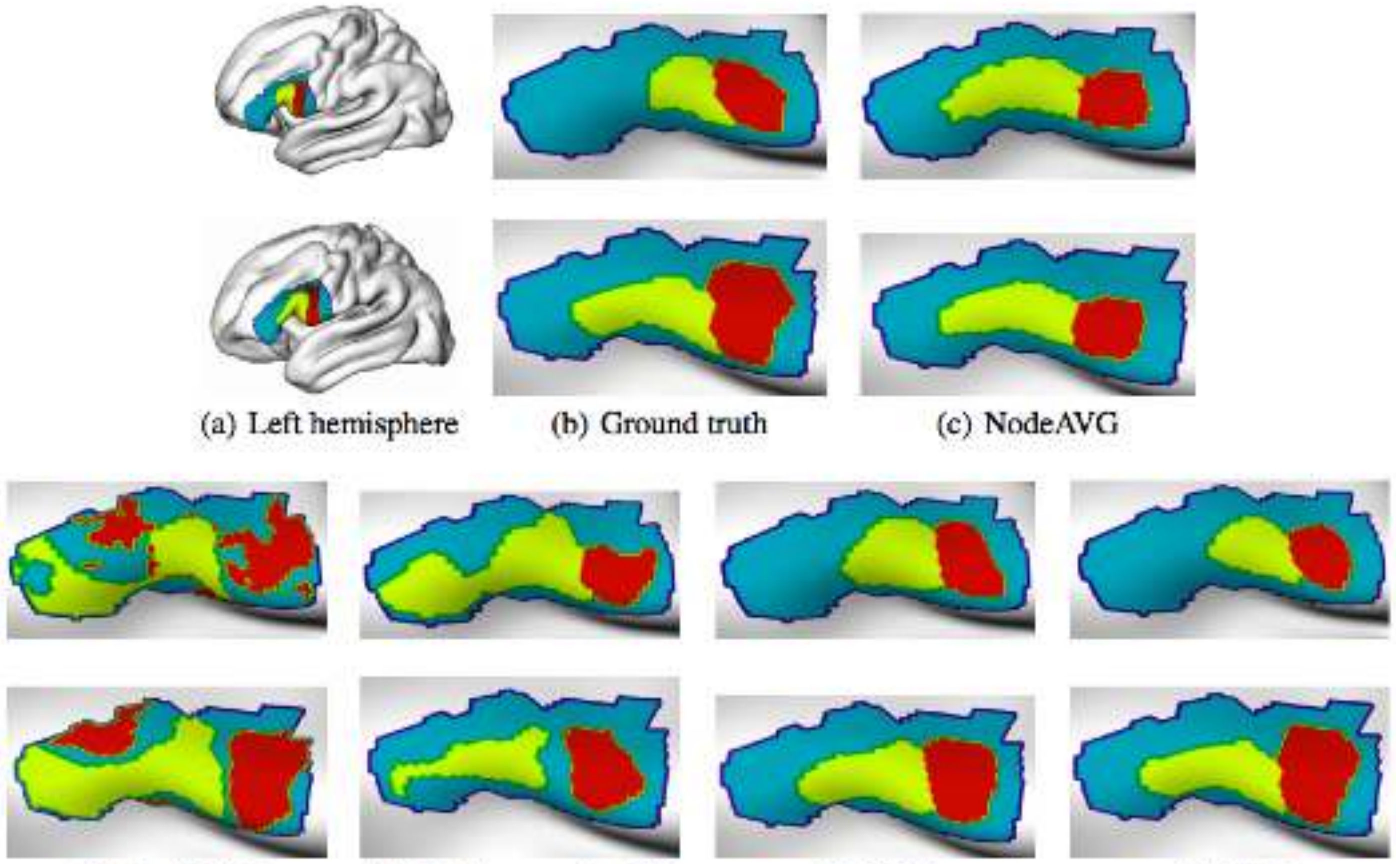
	PPI	HCP
# Graphs	24	100
# Nodes	56944	119500
# Edges	818716	342900
# Features / Node	50	9
# Classes	121 (multilabel)	3
# Training Nodes	44906 (20 graphs)	95600 (80 graphs)
# Validation Nodes	6514 (2 graphs)	11950 (10 graphs)
# Test Nodes	5524 (2 graphs)	11950 (10 graphs)

Meshes: one per subject
Features: structural features, functional features
Output: area 44, area 45, none

RESULTS PPI


F1-score [%]
100
90
80
70
60
50
40
30
20
10
0
39.6
42.2
50
59.8
60
61.2
76.8
93.4
97.3
Random
MLP
GraphSAGE-GCN
GraphSAGE-mean
GraphSAGE-pool
GraphSAGE-LSTM
GraphSAGE (ours)
Const-GAT
GAT

RESULTS CORTICAL MESHES



(a) Left hemisphere (b) Ground truth (c) NodeAVG

(d) NodeMLP (e) Jakobsen et al. [21] (f) GCN (g) GAT

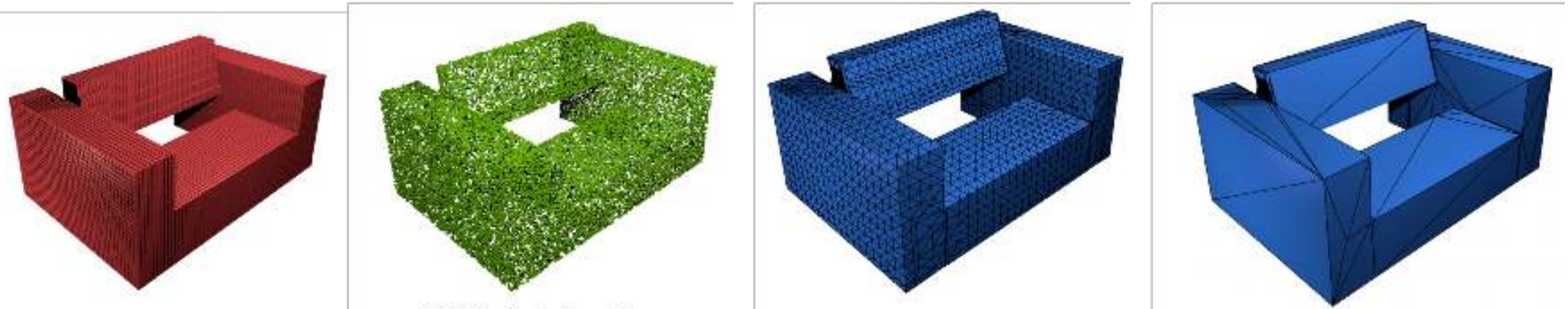
(Cucurull et al., 2018)

GRAPHS WRAP UP

- We introduced GAT, a model which:
  - Implicitly allows assigning different weights to nodes in a same neighborhood.
  - Provides a trivially localized operator.
  - Does not depend on upfront access to the graph structure (inductive problems).
  - Has a fixed number of parameter (dependent on the feature count, not on node/degree counts).
  - Computationally efficient: self-attention parallelizable across edges, computation of output features parallelizable across nodes.
- We highlighted its potential in 3 citation network datasets as well as 2 applications in the biomedical domain.

OTHER GRAPH APPLICATIONS

3D UNDERSTANDING



(a) Voxels (262, 144 units)

(b) Point cloud (30, 000 points)

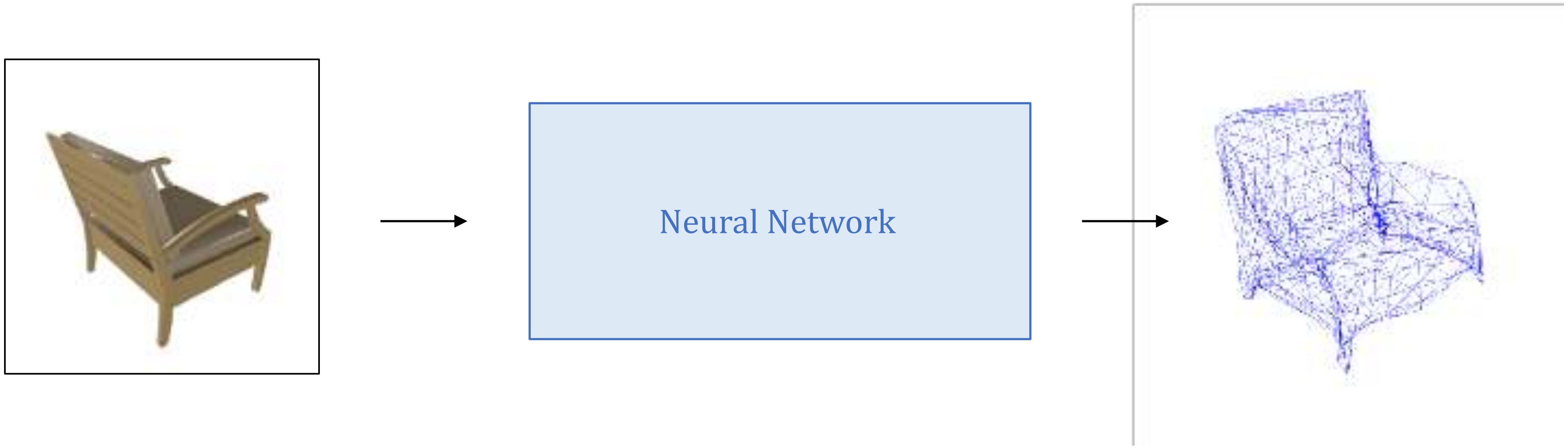
(c) Uniform mesh (2416 vertices)

(d) Adaptive mesh (120 vertices)

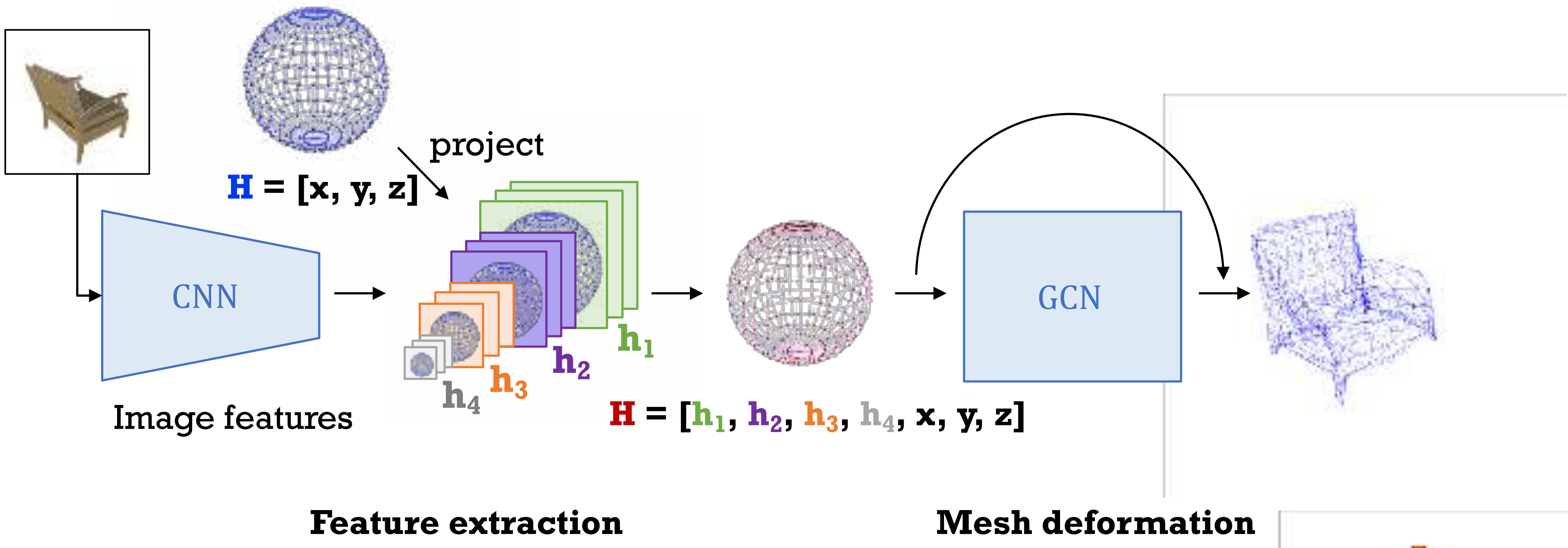
Figure credit: Edward Smith

3D UNDERSTANDING TASK

3D reconstruction from a single view image.



UNIFORM MESH RECONSTRUCTION



Trained to minimize the Chamfer loss:

$$\sum_{p\in S}\min_{q\in\hat{S}}\|p-q\|_2^2+\sum_{q\in\hat{S}}\min_{p\in S}\|p-q\|_2^2$$

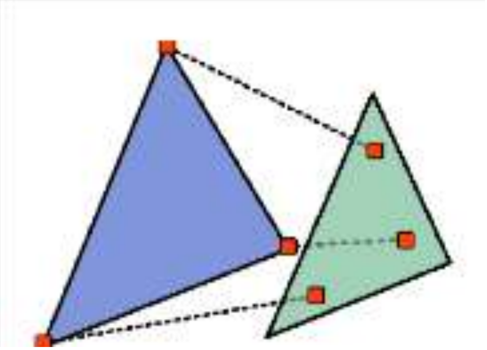
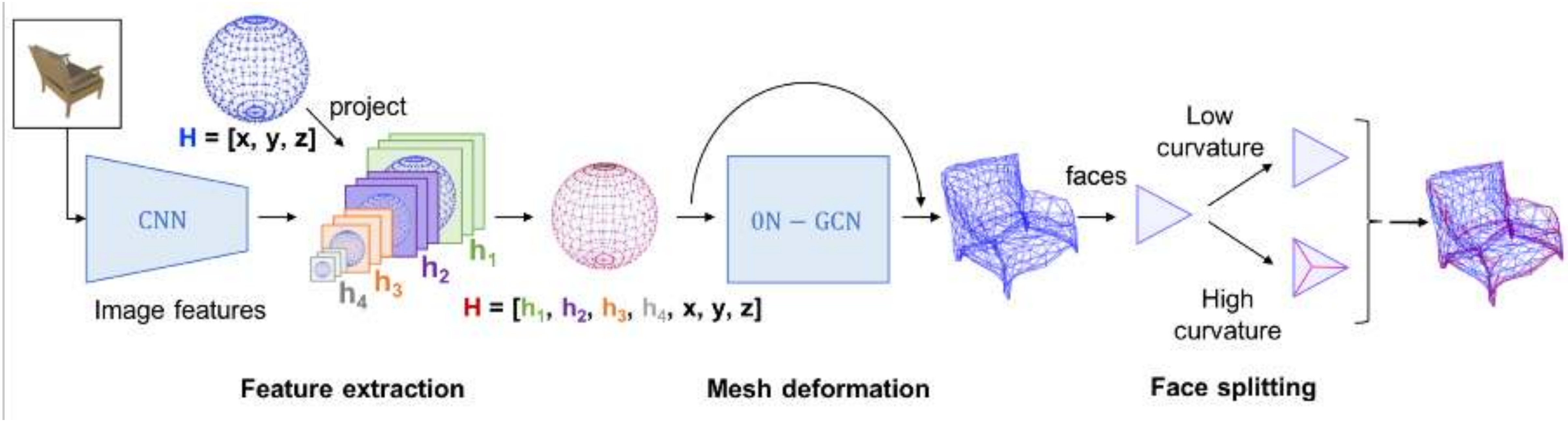


Figure credit: Edward Smith

ADAPTIVE MESH RECONSTRUCTION



Trained with:

- Point-to-surface loss, accounting for the surface defined by the vertices, rather than their position.
- Global mesh loss, comparing embeddings of the ground truth mesh vs the predicted one.

(Smith et al., 2019)

ADAPTIVE MESH RESULTS (1)

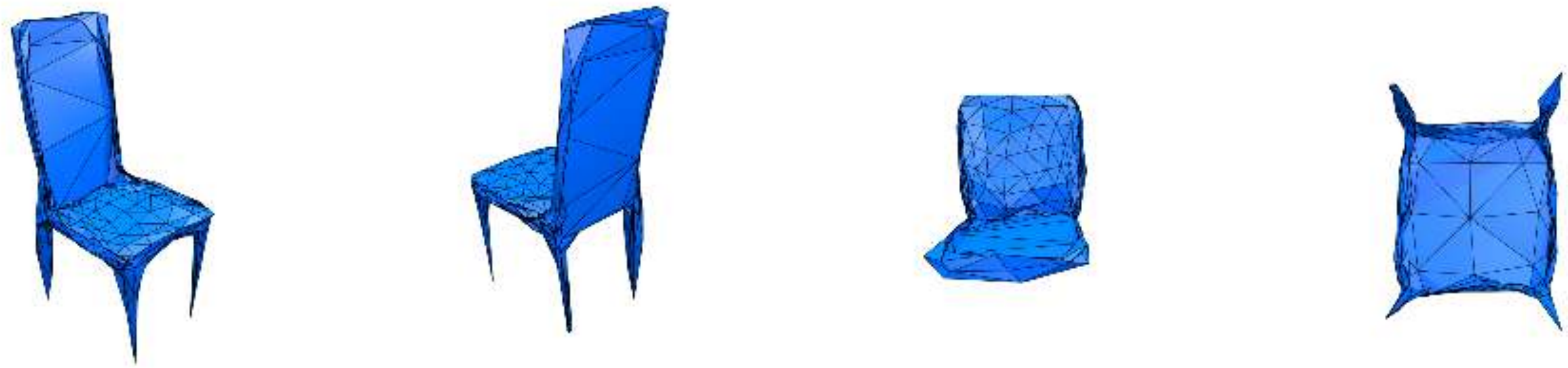


Figure credit: Edward Smith

(Smith et al., 2019)

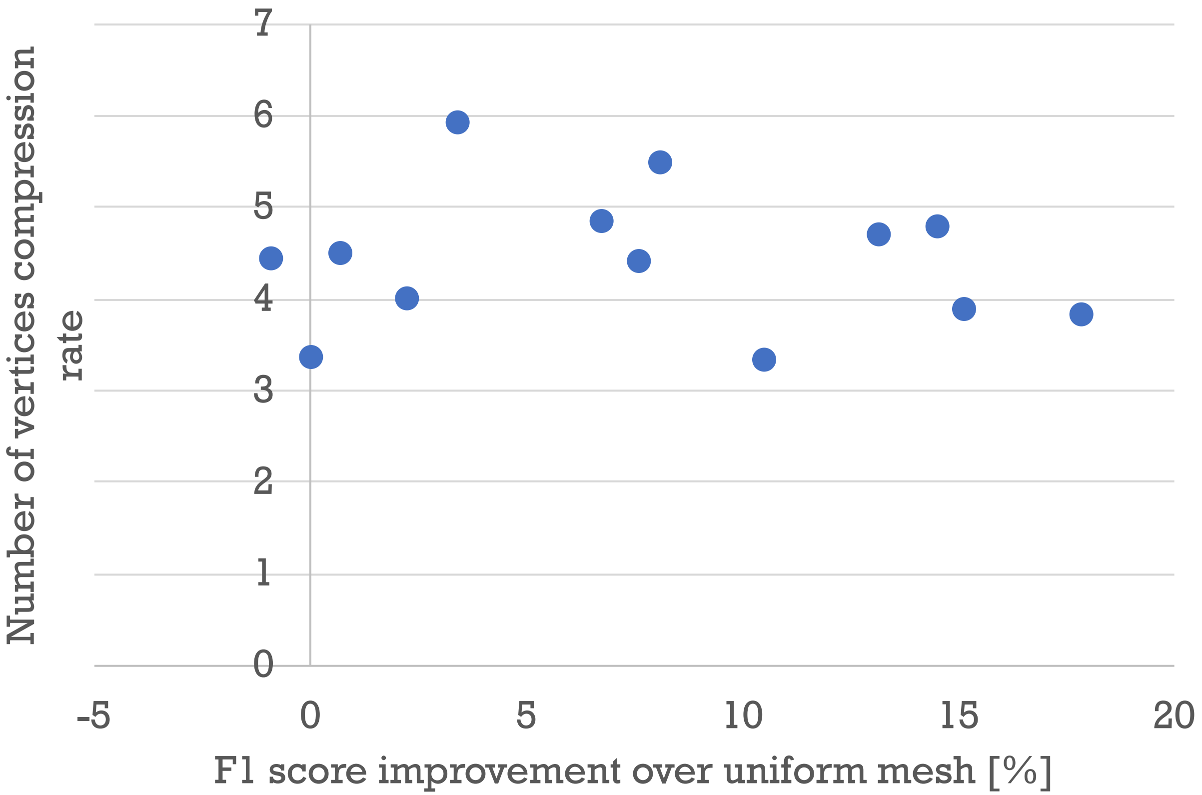
ADAPTIVE MESH RESULTS (2)

Input image | (Smith et al., 2019) | (Wang et al., 2018)

Input image | (Smith et al., 2019) | (Wang et al., 2018)

Figure credit: Edward Smith

ADAPTIVE MESH RESULTS (3)



https://arxiv.org/pdf/1901.11461.pdf

SET PREDICTION

L. Pineda*, A. Salvador*, M. Drozdzal, A. Romero

*: equal contribution

Soon on arXiv!

MOTIVATION (1)

Among image understanding tasks, image classification has arguably received the most attention.

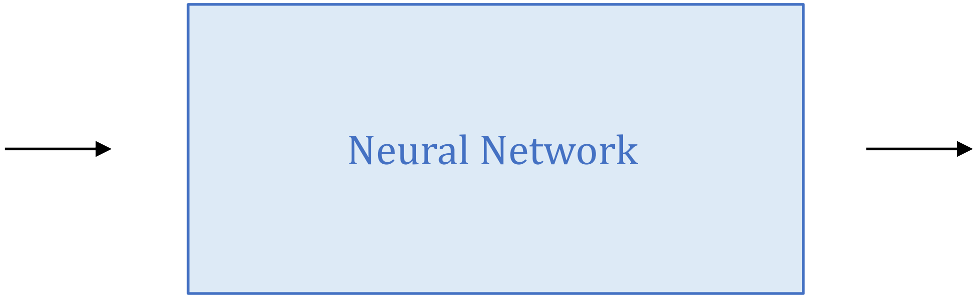
http://image-net.org/challenges/talks_2017/ILSVRC2017_overview.pdf

MOTIVATION (2)

However… everyday life pictures are typically complex scenes, with multiple labels per image.

dog, person, chair, etc	bottle, chair, person, table, etc	sky, tree, building, car, tree, fence, bench, etc	clouds, sky, sunset	ham, tomato, bagel, vinaigrette

MULTI-LABEL CLASSIFICATION



ham,
tomato,
bagel,
vinaigrette

Framed as an image-to-set (of labels) prediction problem:

- Image labels may exhibit relevant dependencies (co-occurrences)
- The number of labels per image is variable
- Order of labels should not matter

DL SET PREDICTION MODELS (1)

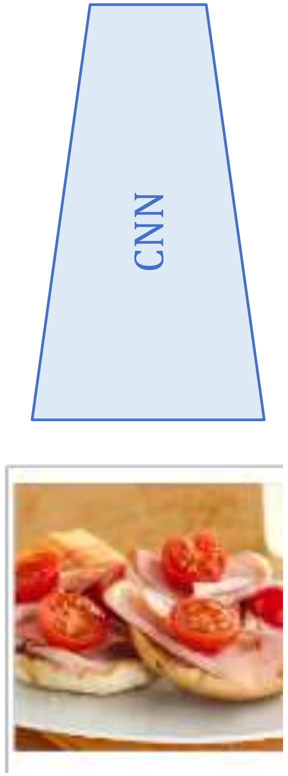


Image feature extractor

Figure credit: Amaia Salvador

DL SET PREDICTION MODELS (2)

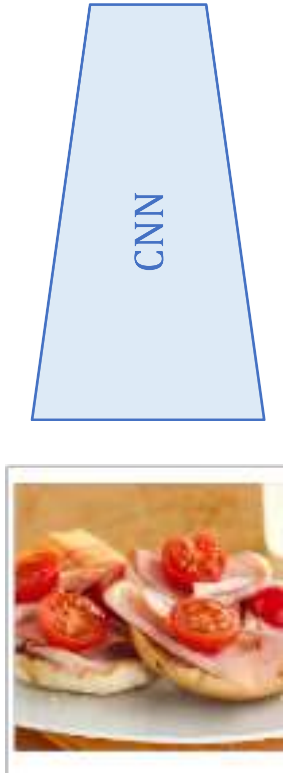
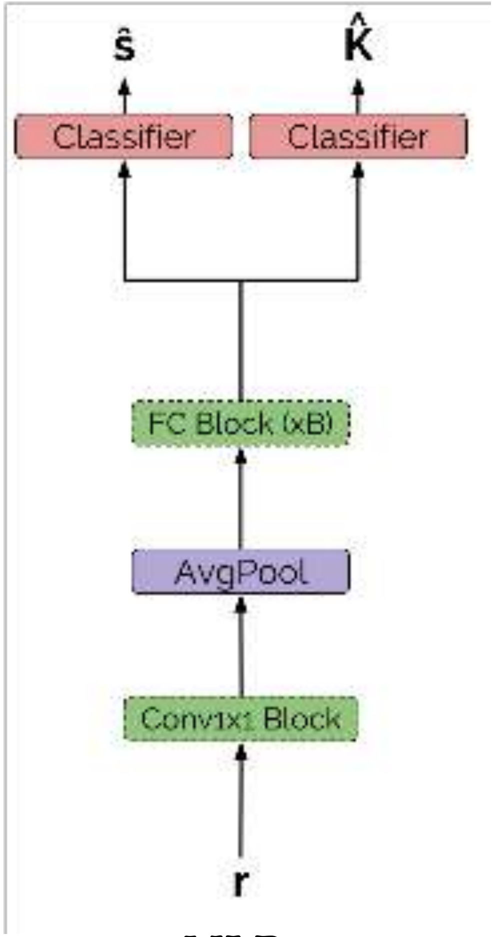


Image feature extractor



MLP

Figure credit: Amaia Salvador

DL SET PREDICTION MODELS (3)

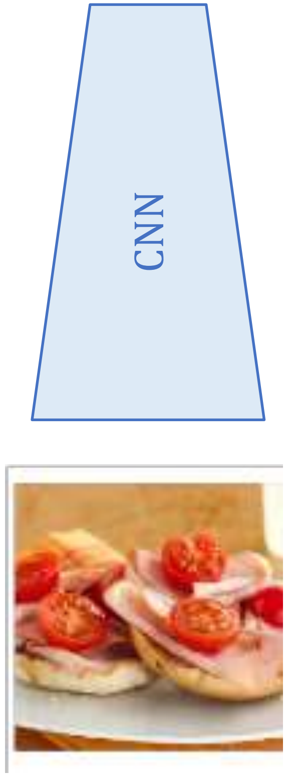
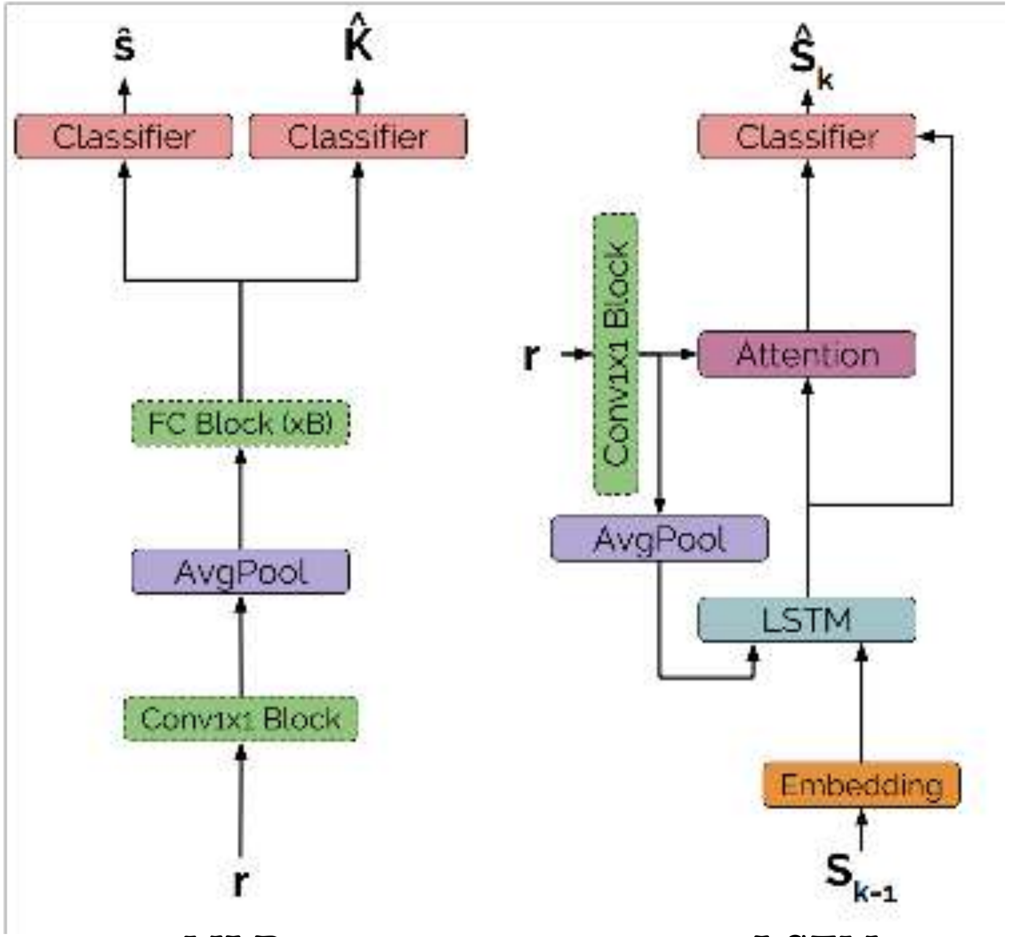


Image feature extractor



MLP

LSTM

Figure credit: Amaia Salvador

DL SET PREDICTION MODELS (4)

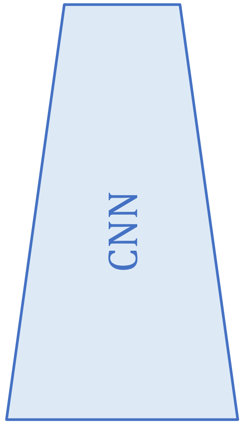
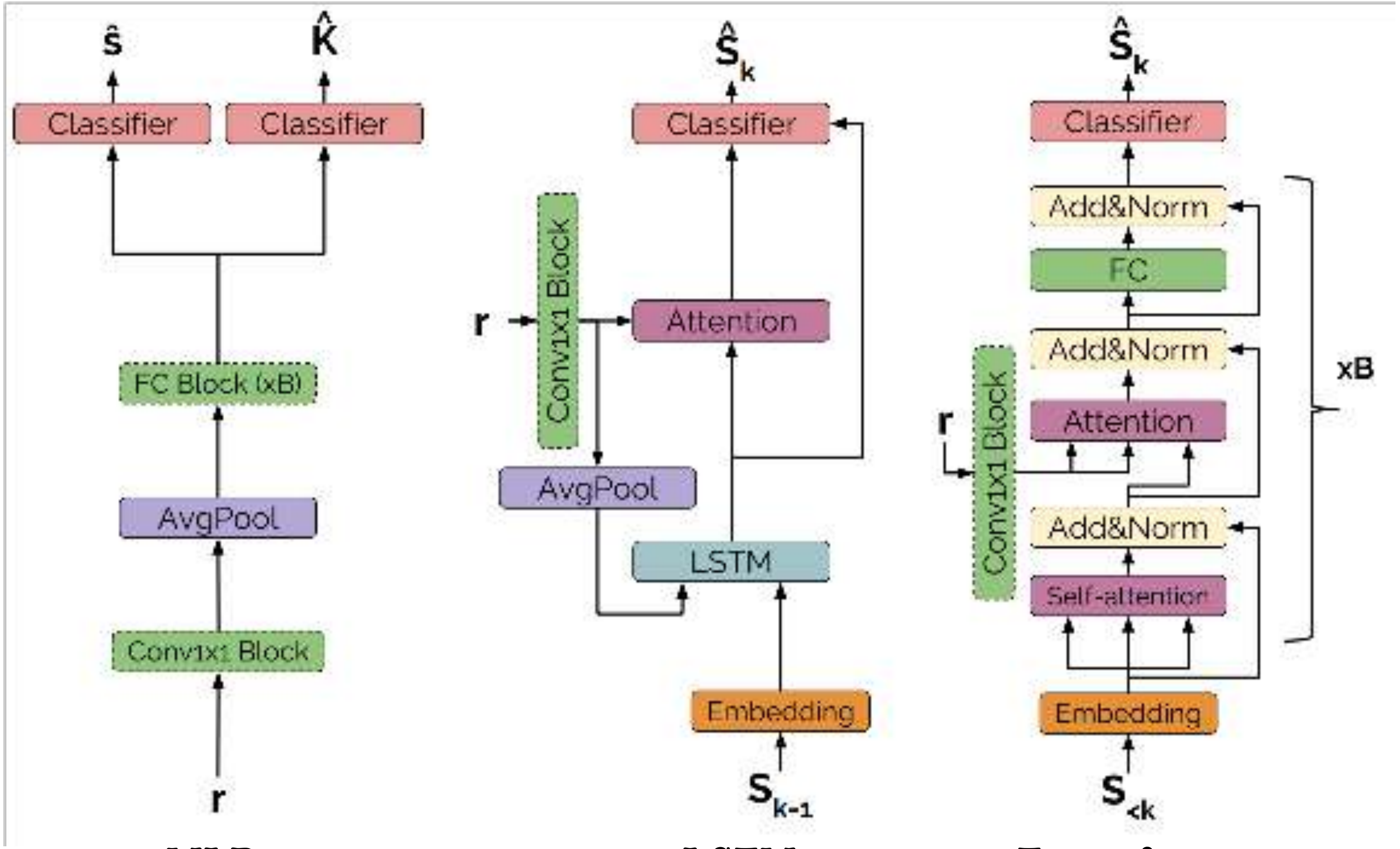


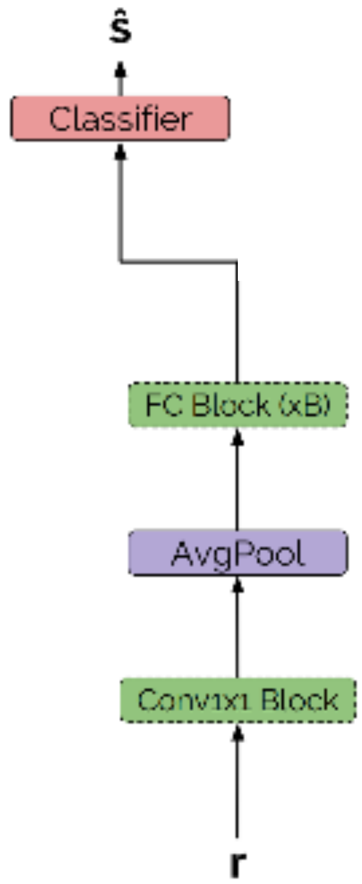
Image feature extractor



MLP LSTM Transformer

Figure credit: Amaia Salvador

MLP: PREDICTION LOSSES (1)



MLP

Figure credit: Amaia Salvador

If we consider independent outputs:

- ✓ $\hat{\mathbf{s}}$ represents a vector of independent Bernoulli distributions (sigmoid non-linearity)
- ✓ We could use binary cross-entropy as loss function:

$$-\frac{1}{L}\sum_l \mathbf{s}_l \log \hat{\mathbf{s}}_l + (1-\mathbf{s}_l)\log(1-\hat{\mathbf{s}}_l),$$

where $\mathbf{s}_l$ is the ground truth.

- ✓ We could use a structured loss (soft IoU):

$$\frac{\sum_l \mathbf{s}_l\, \hat{\mathbf{s}}_l}{\sum_l \mathbf{s}_l + \hat{\mathbf{s}}_l \; - \mathbf{s}_l\hat{\mathbf{s}}_l}$$

MLP: PREDICTION LOSSES (2)

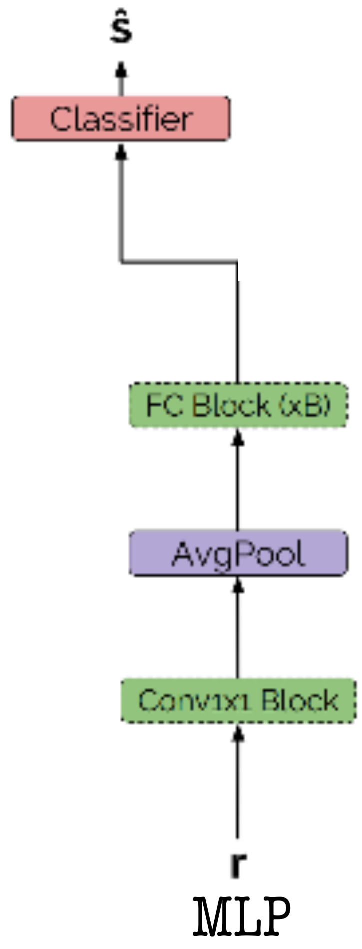


Figure credit: Amaia Salvador

If we consider dependent outputs:

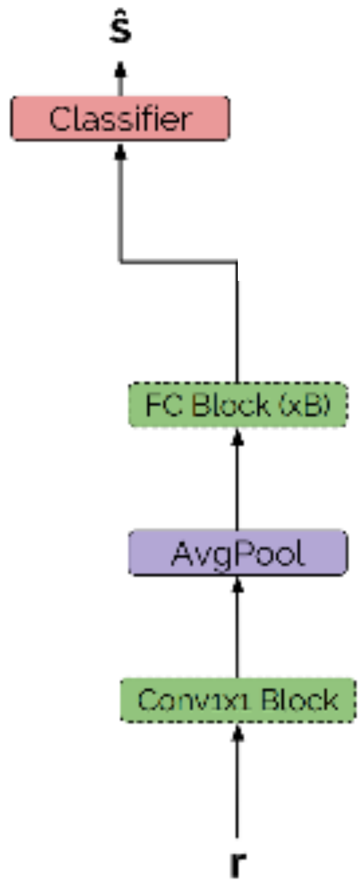
- ✓ $\hat{\mathbf{s}}$ represents a categorical distribution (softmax non-linearity)
- ✓ What could we use as a target?

  If the ground truth is [1, 0, 0, 1, 0], we could use [0.5, 0, 0, 0.5, 0] as $\mathbf{s}$

- ✓ Then, we could use cross-entropy as proxy loss function:

$$-\frac{1}{L}\sum_{l} \mathbf{s}_l \log \hat{\mathbf{s}}_l$$

MLP: CARDINALITY PREDICTION



MLP

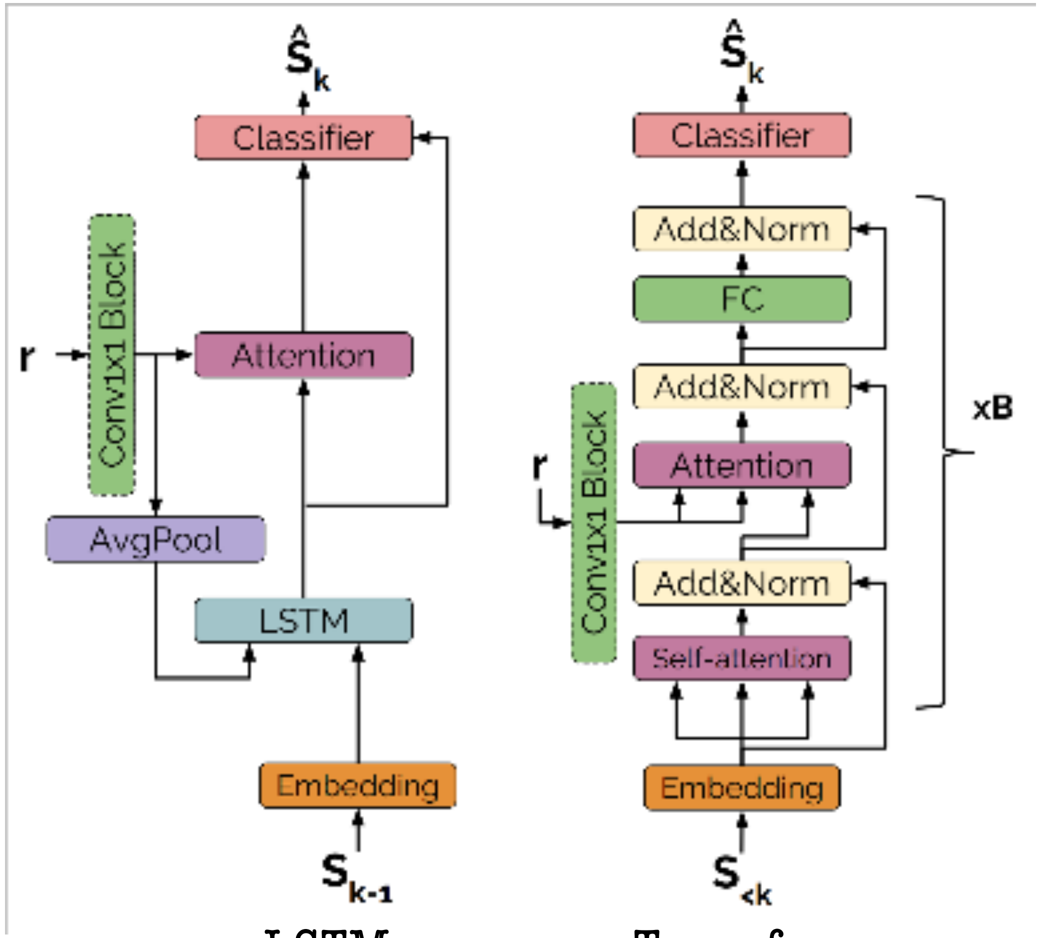
Figure credit: Amaia Salvador

- ✓ We could use $\hat{\mathbf{s}}$ to determine the cardinality of the set:

$$\hat{\mathbf{s}}_l > 0.5 \; \forall l$$

- ✓ We could predict the cardinality from a second output:
  - ➢ MSE (ReLU non-linearity)
  - ➢ Categorical cross-entropy (softmax non-linearity)

AUTO-REGRESSIVES: PREDICTION LOSSES (1)



LSTM Transformer

If we consider ordered outputs:

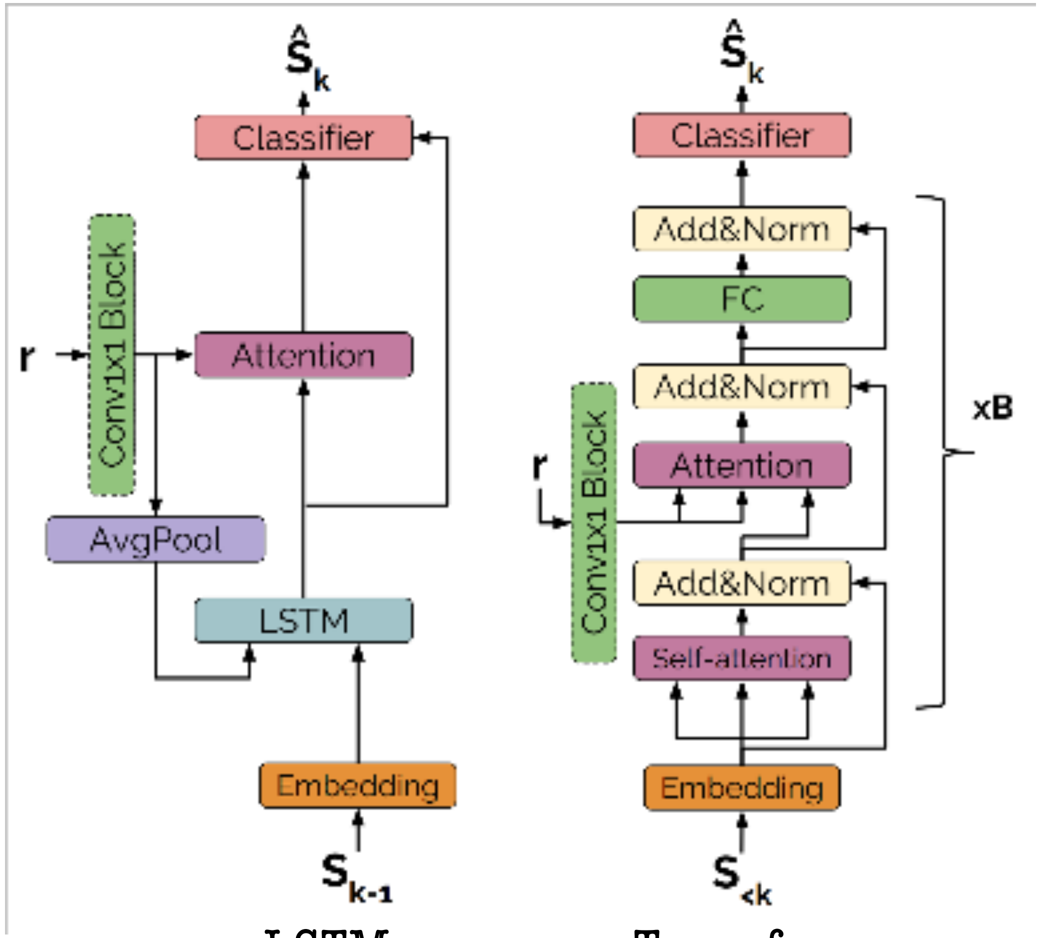
- ✓ $\hat{\mathbf{s}}_k$ (at time step k) represents a categorical distribution (softmax non-linearity)
- ✓ We could use categorial cross-entropy as loss function for each time step:

$$-\frac{1}{L}\sum_{l} \mathbf{s}_k \log \hat{\mathbf{s}}_k \,,$$

where $\mathbf{s}_k$ is a one hot vector containing one of the ground truth labels.

Figure credit: Amaia Salvador

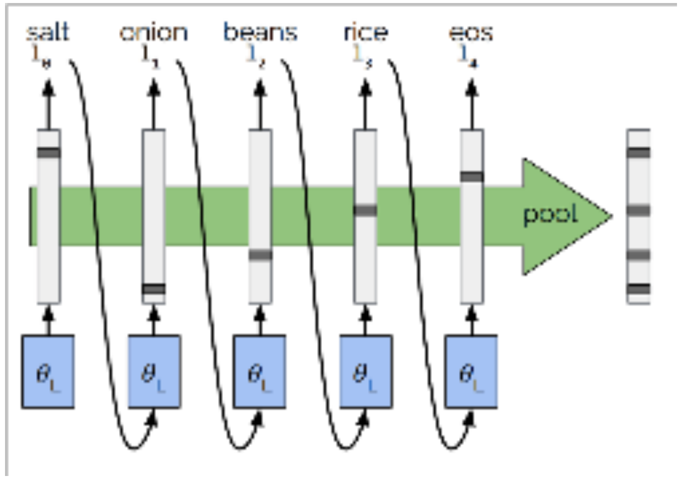
AUTO-REGRESSIVES: PREDICTION LOSSES (2)



LSTM Transformer

If we consider no order:

✓ Pool across time steps:



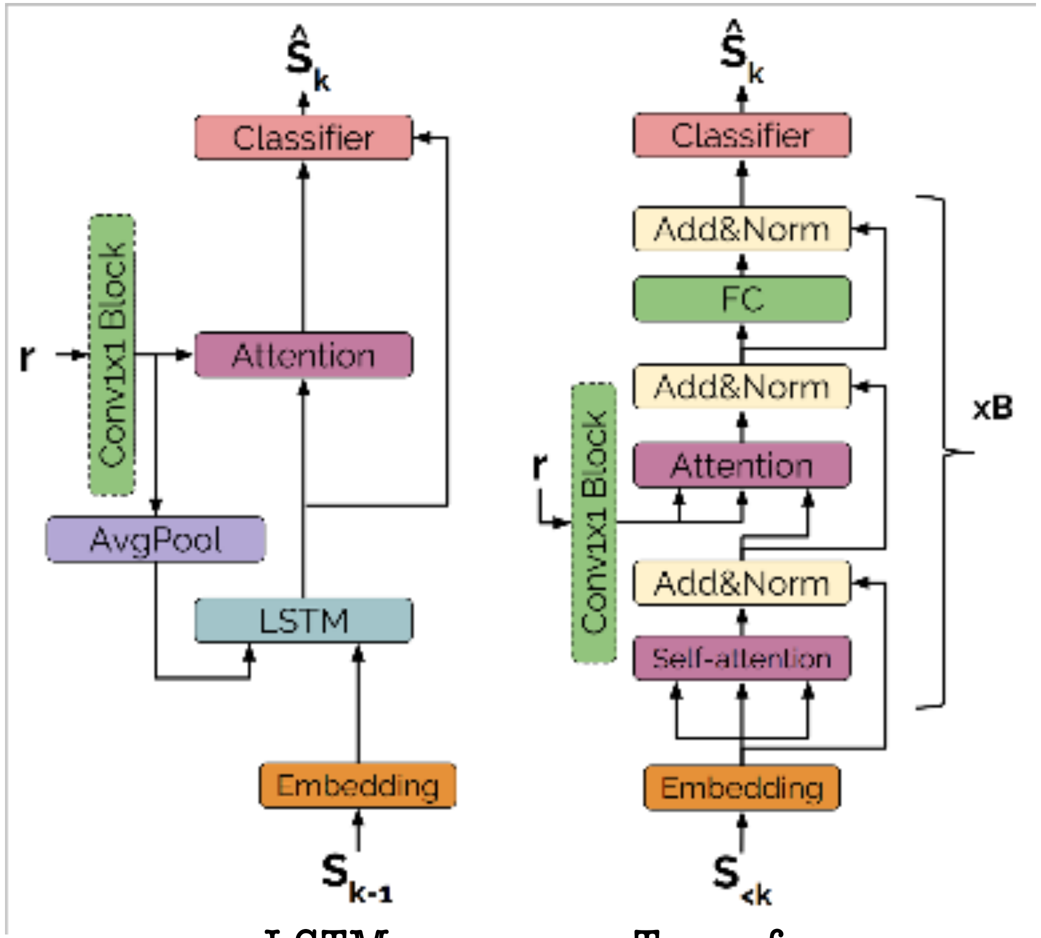
✓ We could use binary cross-entropy as loss function

$$-\frac{1}{L}\sum_{l} \mathbf{s}_l \log \hat{\mathbf{s}}_l + (1-\mathbf{s}_l)\log(1-\hat{\mathbf{s}}_l),$$

where $\mathbf{s}_l$ is the ground truth

Figures credit: Amaia Salvador

AUTO-REGRESSIVES: CARDINALITY PREDICTION



LSTM Transformer

If we consider ordered outputs:

- ✓ end-of-sequence (*eos*) token predicted as last step
- ✓ *eos* step included in categorical cross-entropy loss

If we consider no ordered:

- ✓ *eos* token predicted as last step
- ✓ Additional loss at each time step to decide whether *eos* should be predicted (categorial cross-entropy)

Figure credit: Amaia Salvador

HOW DO ALL THESE METHODS COMPARE?

DATASETS

VOC 2007
20 categories
4.9k train images
512 valid images
4.9k test images
All (partially) visible objects

MS COCO 2014
80 categories
74.5k train images
8.3k valid images
40.5k test images
All (partially) visible objects

ADE20k
150 categories
18.2k train images
2k valid images
2k test images
All (partially) visible objects

NUS-WIDE
81 categories
45.6k train images
16.2k valid images
107.9k test images

Recipe1M
1486 categories
252.5k train images
5k valid images
54.5k test images

RESULTS (1)

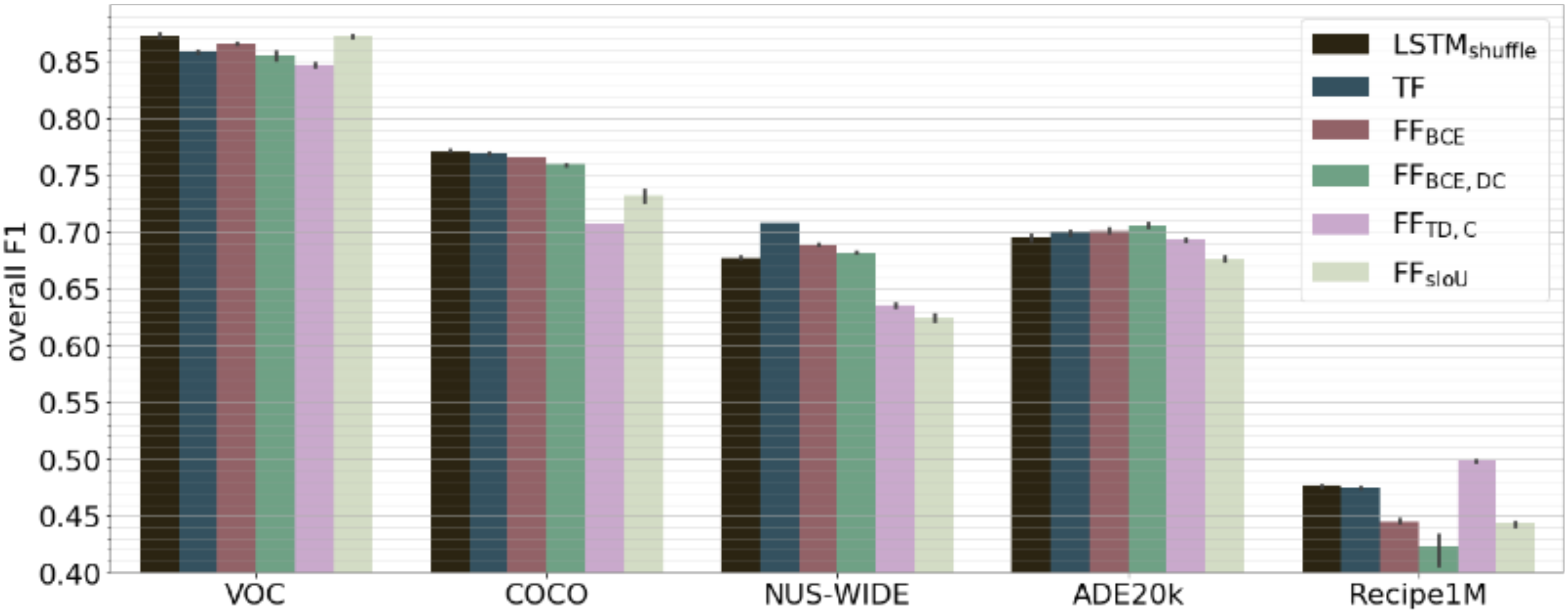


Figure credit: Luis Pineda

RESULTS (2)

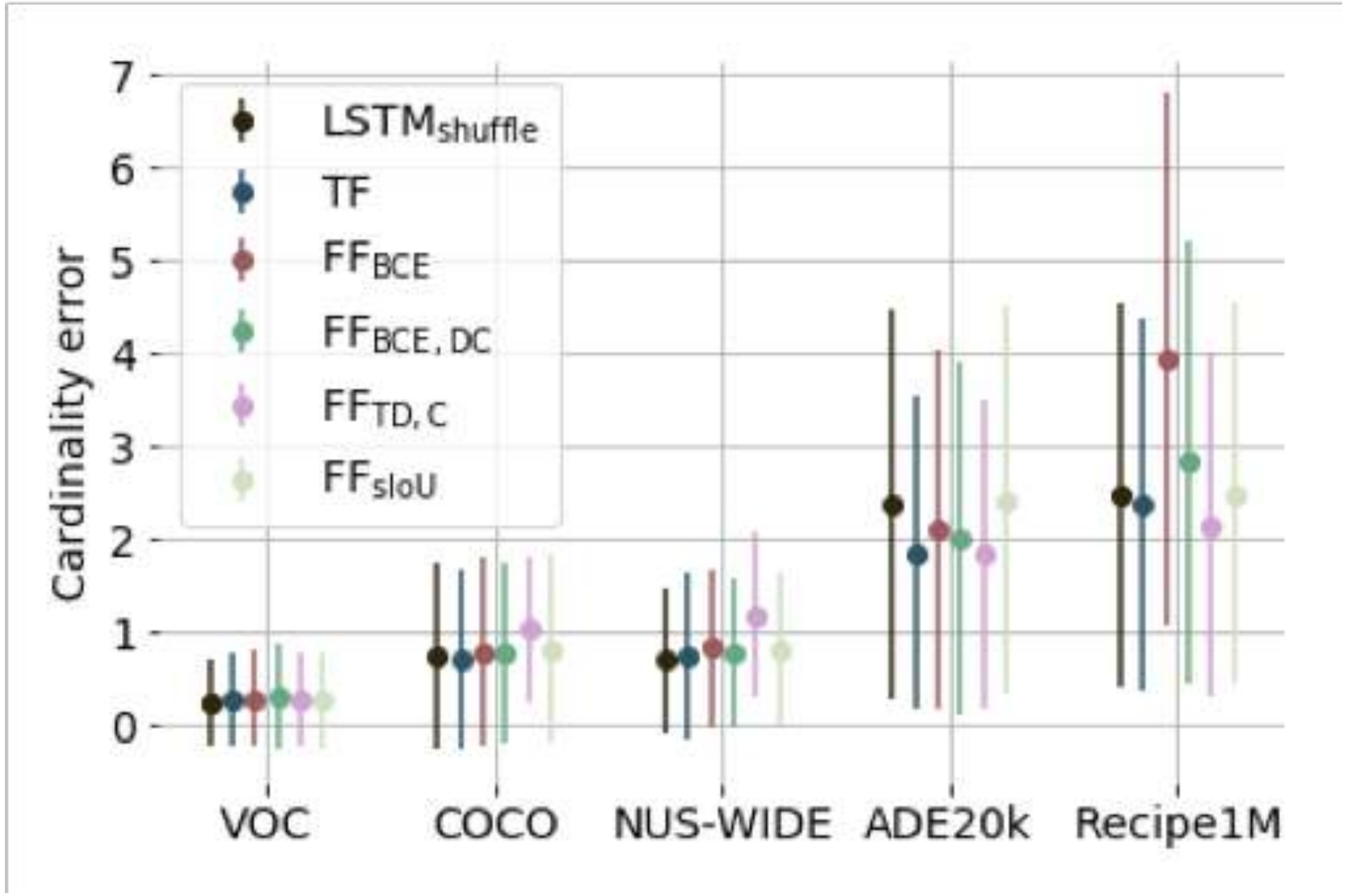


Figure credit: Luis Pineda

SETS WRAP UP

- We presented a comprehensive analysis of methods suitable for image-to-set prediction:
  - We evaluated their performance in 5 diverse datasets
  - Using a uniform set of metrics
  - Comparable budgets for hyperparameter tuning
- Our analysis suggests that auto-regressive models are better choices than feed-forward models for the task, performing consistently well across all considered datasets.
  - They inherently handle set cardinality prediction, label co-occurrences.
  - Shuffling label order tends to increase performance.

RECIPE GENERATION

A. Salvador, M. Drozdzal, X. Giro-Nieto, A. Romero

Inverse Cooking: Recipe Generation from Food Images @ CVPR 2019

Based on A. Salvador's slides

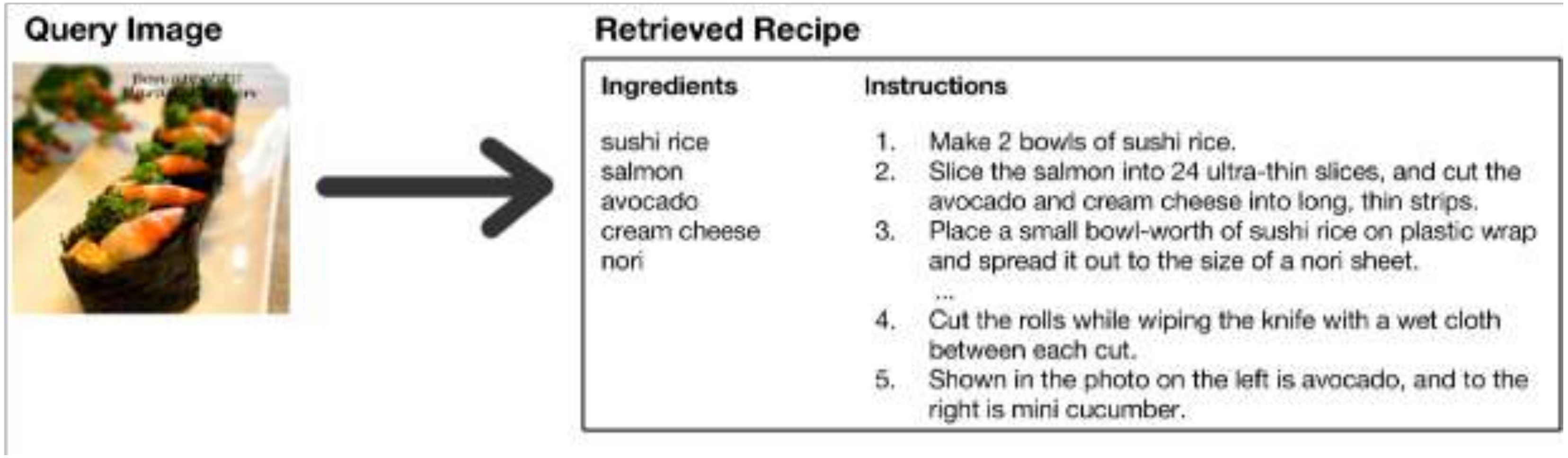
MOTIVATION

Shapes and colors

Invisible ingredients

More abstract understanding of what food is.

WHY GENERATING RECIPES?



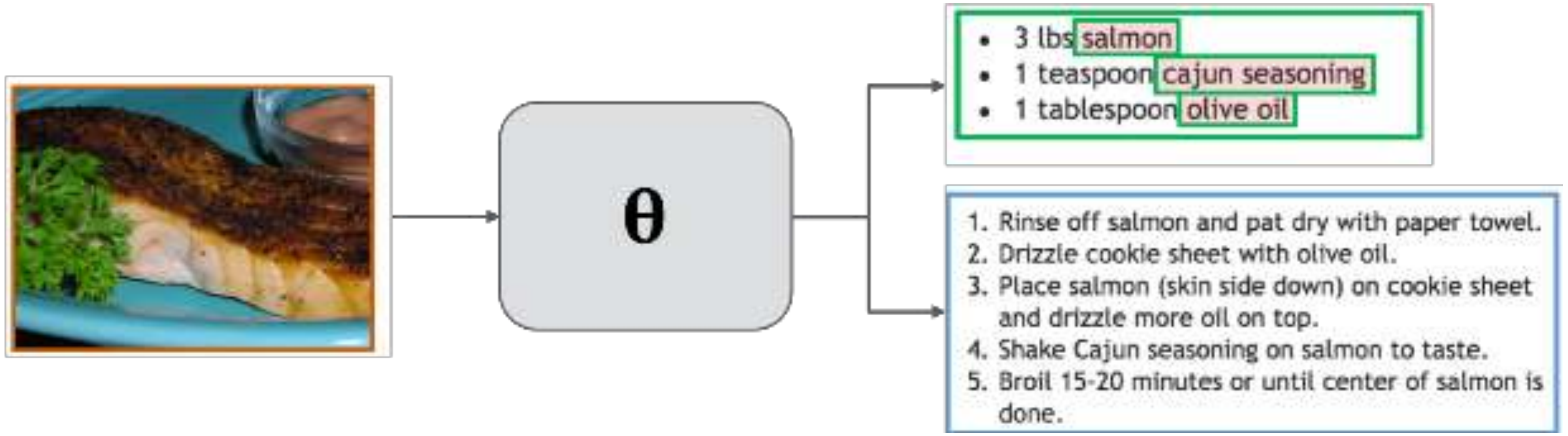
Previous work focuses on retrieving recipes for a query image. But...

- Retrieval is constrained to database recipes (inaccurate matches).
- Framed to retrieve ingredients and recipes as a whole.
- Prohibits user manipulation (e.g. ingredient replacement).

Figure credit: Amaia Salvador

PROBLEM FORMULATION

Generate ingredients list & cooking instructions.



Ingredient's list treated as sets.

- Order invariance.
- Variable cardinality.
- Dependencies among different elements in the set.

Figure credit: Amaia Salvador

RECIPE GENERATION FROM IMAGE AND INGREDIENTS' SET

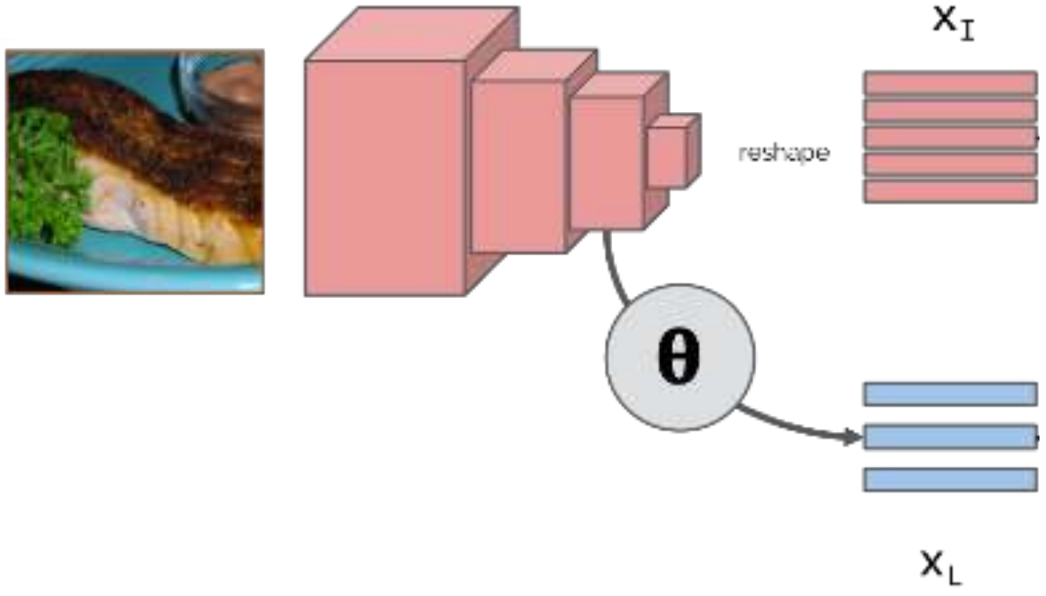
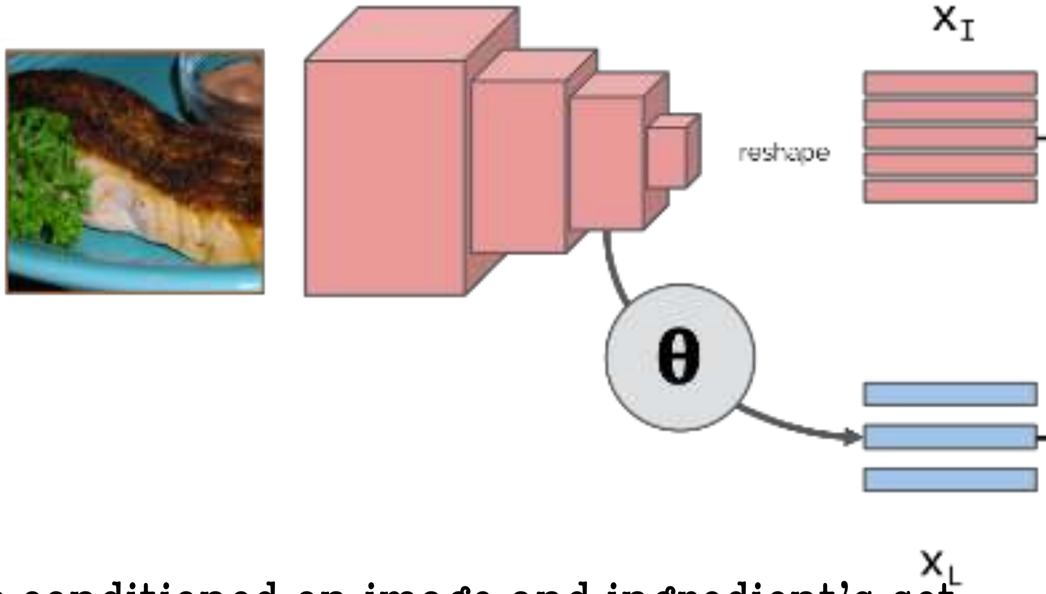
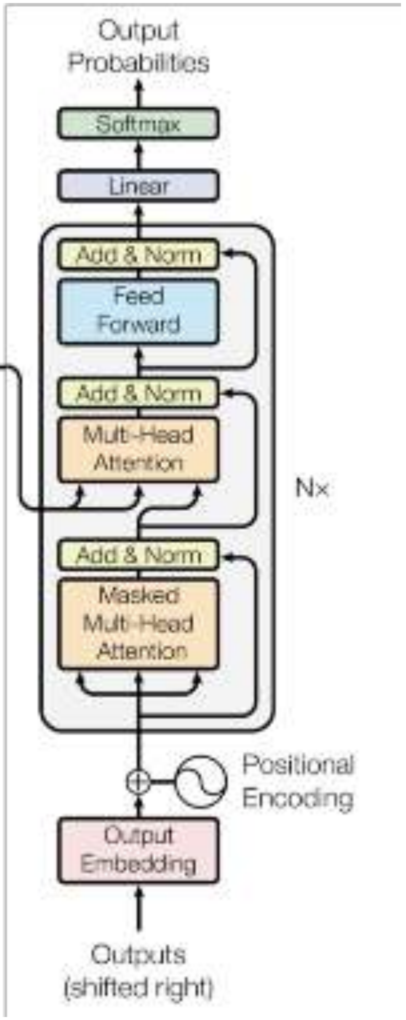


Figure credit: Amaia Salvador

RECIPE GENERATION FROM IMAGE AND INGREDIENTS' SET



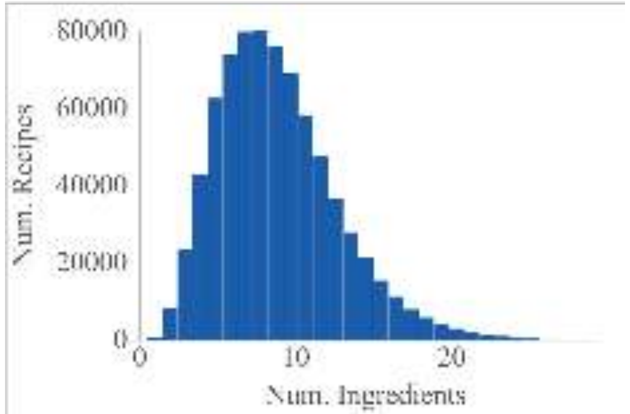


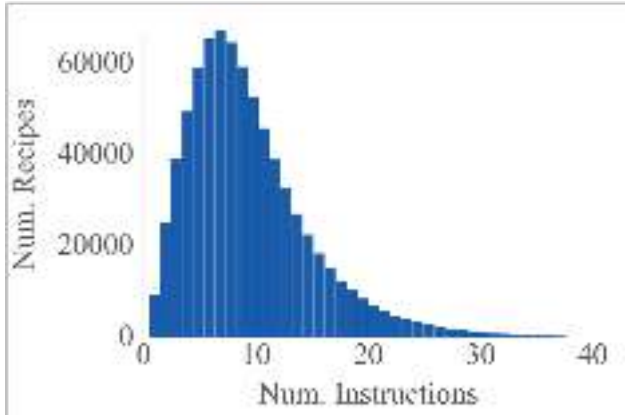
Transformer conditioned on image and ingredient's set.
Positional encoding.
Trained with NLL (penalizing order).

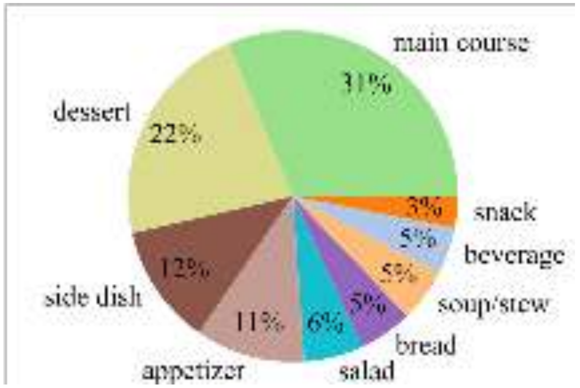
Figure credit: Amaia Salvador

DATASET: RECIPE 1M

A structured dataset of 1M recipes and 800k images scraped from ~20 cooking recipe websites.







The average cooking recipe in our dataset contains 9 ingredients which are transformed over the course of 10 instructions.

Salvador, Hynes et al., 2017

Figure credit: Amaia Salvador

QUALITATIVE RESULTS (1)

beef, onion, tomato, water, pasta, beans, cheese, chili

tomato, onion, beans, pepper, broth, clove, carrot, oil, salt, chili

sugar, flour, salt, baking_soda, butter, egg, extract

egg, flour, sugar, butter, salt, cinnamon, baking_powder, milk

edamame, pepper, beans, corn, dill, juice, italian_dressing, avocado

beans, corn, pepper, tomato, onion, vinegar, oil

Figure credit: Amaia Salvador

QUALITATIVE RESULTS (2)

Title: Basic sugar cookies

Ingredients

- flour
- egg
- butter
- salt
- sugar

Instructions

1. Preheat oven to 350.
2. In a mixing bowl cream butter with sugar.
3. Add the eggs one at a time.
4. Then add the flour and salt and mix thoroughly.
5. Drop by teaspoonfuls on a baking sheet and bake for 8-10 minutes (or until golden brown).
6. Remove from oven and transfer to a wire rack to cool.

Figure credit: Amaia Salvador

QUALITATIVE RESULTS (3)

Title: Macaroni and cheese with beans

Ingredients

- onion
- pepper
- cheese
- salt
- beans
- clove
- pasta
- bacon
- tomato
- oil
- water

Instructions

1. Cook pasta according to package directions.
2. Meanwhile, heat olive oil in a deep skillet over medium high heat.
3. Saute onion and garlic in olive oil until tender, 7 to 10 minutes.
4. Stir in the beans, bacon, salt and pepper.
5. Saute, stirring, until beans are heated through, 10 to 12 minutes.
6. Stir in water and tomato sauce, bring to a boil, then reduce heat and simmer, covered, until sauce is thickened, about 10 minutes.
7. Stir in cheese.
8. Drain pasta and place on serving plates.

Figure credit: Amaia Salvador

RECIPE GENERATION WRAP UP

We coupled the set prediction module with a conditional transformer, which predicts a recipe from an input image and the predicted ingredients' set, showcasing compelling results.

COLLABORATORS

Guillem Cucurull
Arantxa Casanova

Konrad Wagstyl
Estrid Jakobsen
Alan Evans

Michal Drozdzal
Luis Pineda

Edward Smith
Scott Fujimoto
Dave Meger

Amaia Salvador
Xavier Giró-Nieto

Petar Velickovic
Pietro Lio
Konrad Wagstyl

DEEP LEARNING FOR GRAPHS AND SETS

Adriana Romero

March 26th, 2019 @ UPC School

THANKS!