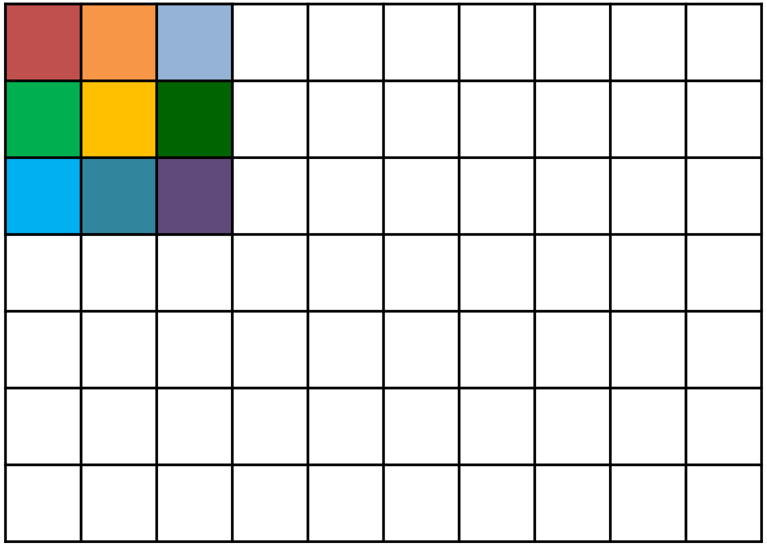
Deep Learning on graphs

The domain so far

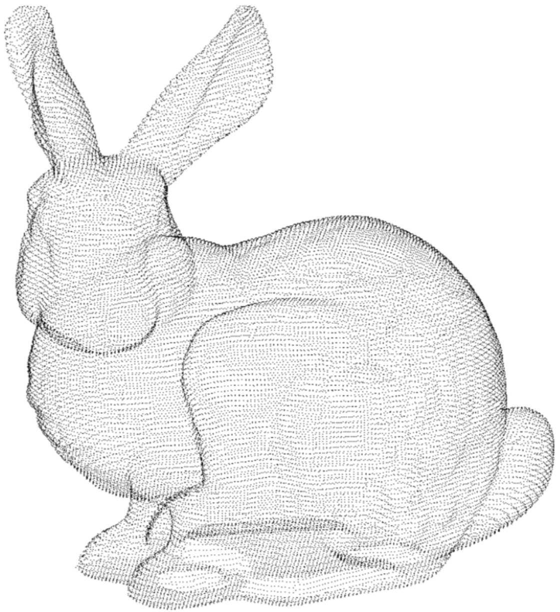
- Regularity on the domain
  - Order of the pixels is important

Your convolution filter imposes a certain structure



A new domain

- Rich Information
  - 3D Point Location
  - Other features:
    - RGB/ Intensity
    - Semantic
- Irregularity
  - Permutation Invariance
  - Transformation Invariance

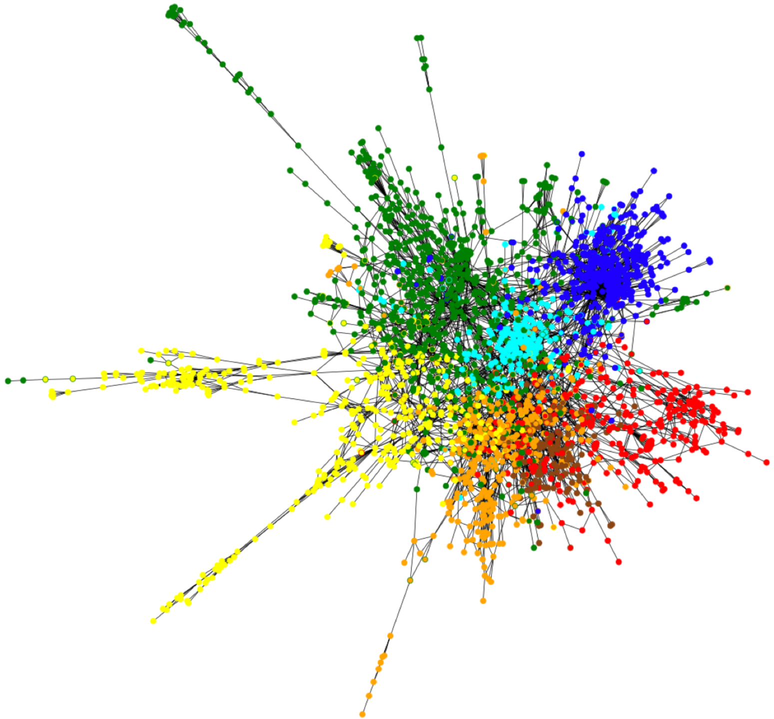


Point Cloud

A new domain

- A citation network
  - Each node is a paper
  - Connection is a citation

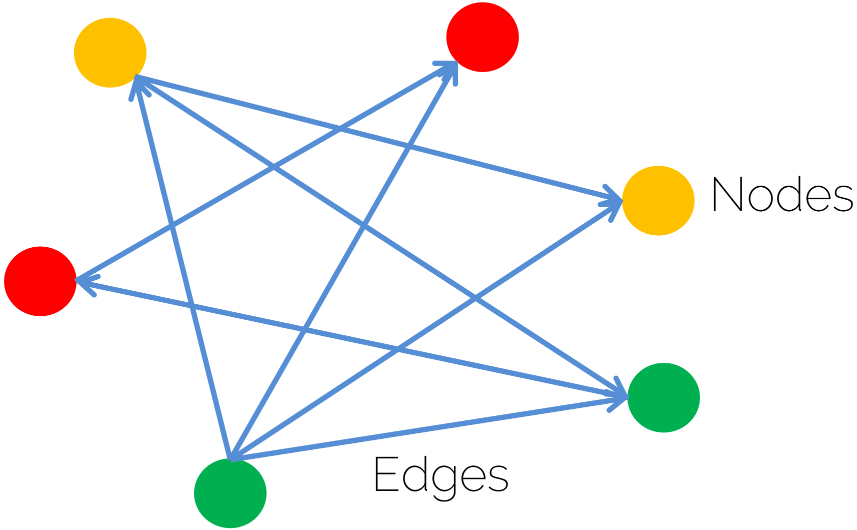
- Similar for:
  - Social networks
  - Recommender systems



M. Bronstein et al. „Geometric deep learning: going beyond Euclidean data". IEEE Signal Processing Magazine. 2017

A graph

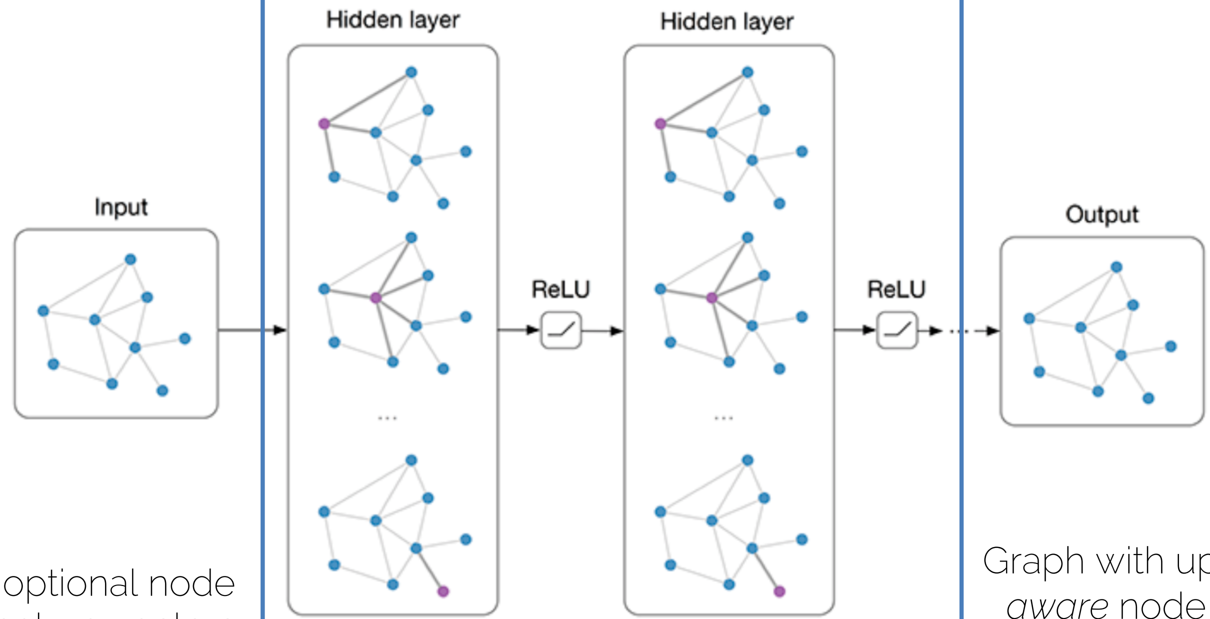
- Node: a concept
- Edge: a connection between concepts



Deep learning on graphs

- Generalizations of neural networks that can operate on graph-structured domains:
  - Scarselli et al. "The Graph Neural Network Model". IEEE Trans. Neur. Net 2009.
  - Kipf et al. "Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2016.
  - Gilmer et al. "Neural Message Passing for Quantum Chemistry". ICML 2017
  - Battaglia et al. "Relational inductive biases, deep learning, and graph networks". arXiv 2018 (review paper)
- Key challenges:
  - Variable sized inputs (number of nodes and edges)
  - Need **invariance to node permutations**

General Idea



Graph with optional node and edge feature vectors

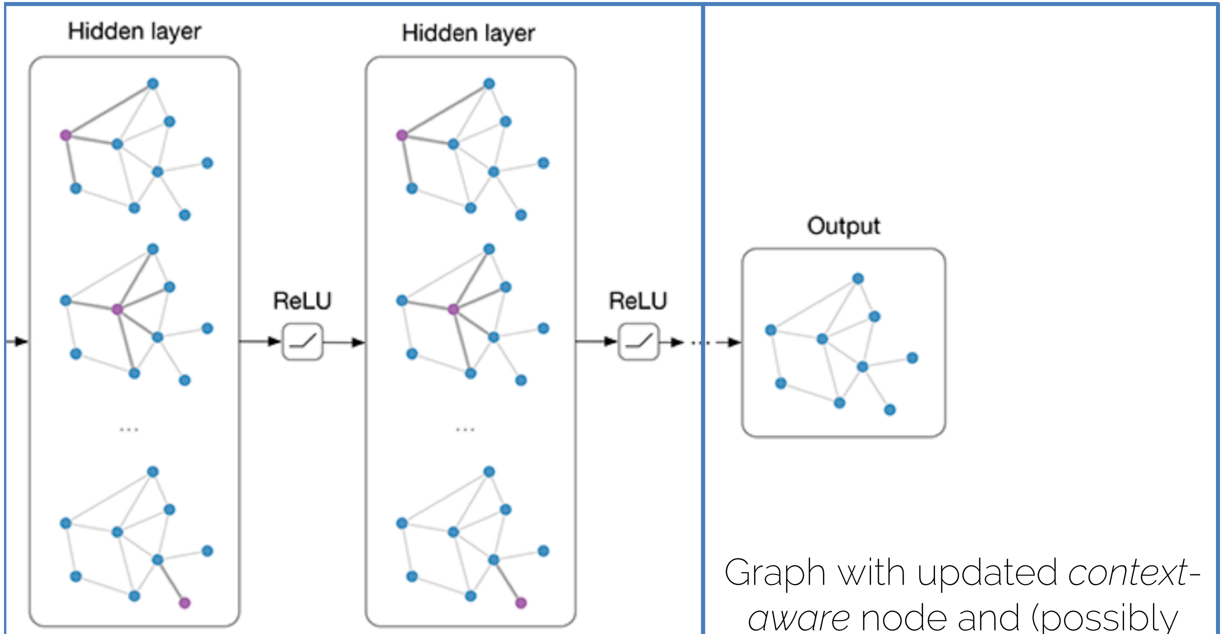
Information propagation across the graph for several iterations

Graph with updated *context-aware* node and (possibly edge) feature vector(s)

Figure credit: https://tkipf.github.io/graph-convolutional-networks/

General Idea

Each update step is understood as a "layer" in common NNs

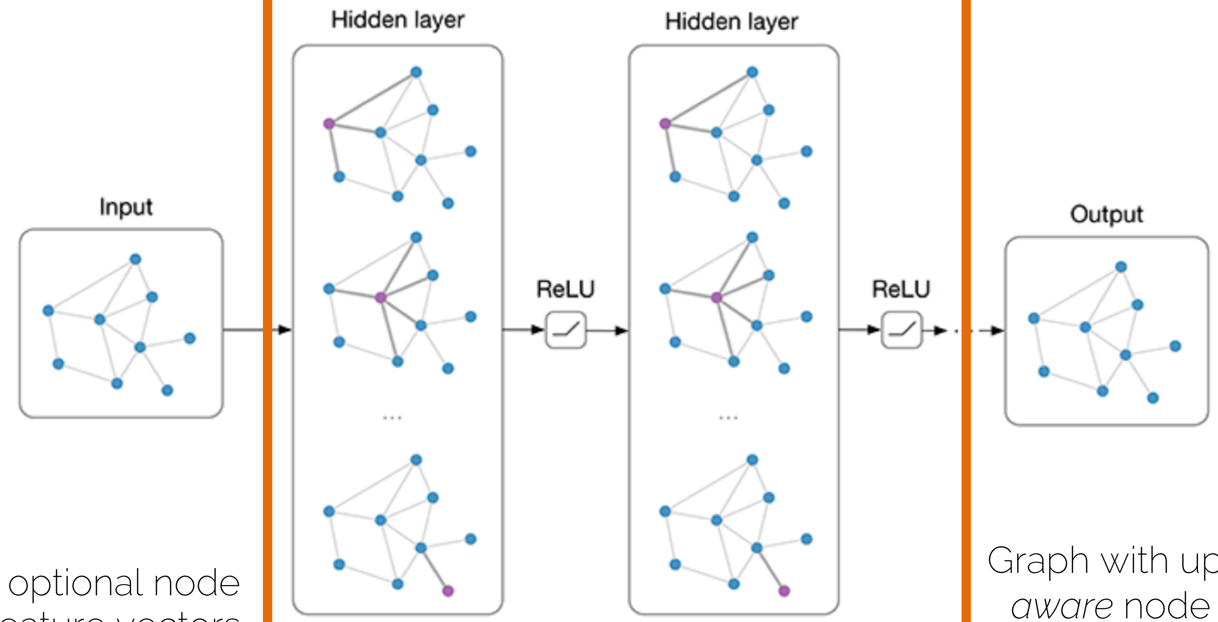


Information propagation across the graph for several iterations

Graph with updated *context-aware* node and (possibly edge) feature vector(s)

Figure credit: https://tkipf.github.io/graph-convolutional-networks/

General Idea



Graph with optional node and edge feature vectors

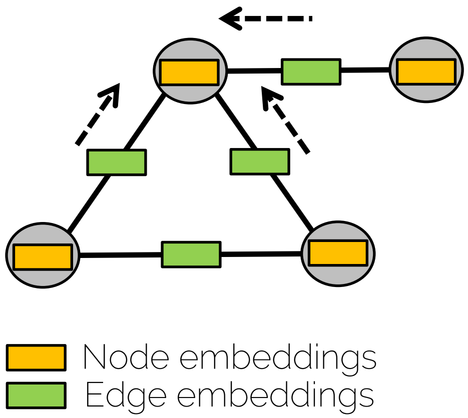
Information propagation across the graph for several iterations

Graph with updated *context-aware* node and (possibly edge) feature vector(s)

Figure credit: https://tkipf.github.io/graph-convolutional-networks/

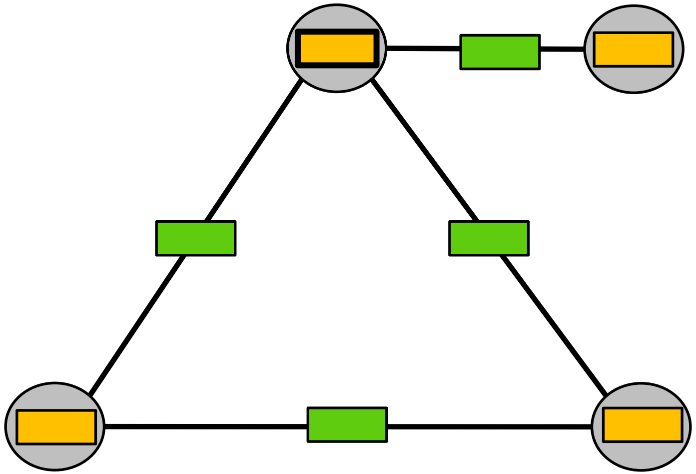
Neural Message Passing

- Notation:
  - Graph: $G = (V, E)$
  - Initial embeddings: $h^{(0)}_{(i,j)}, (i,j) \in E \quad h^{(0)}_i, i \in V$
  - Node embeddings after l steps: $h^{(l)}_i, i \in V$
- Goal:
  - Encode contextual graph information in node embeddings by iteratively combining neighboring nodes' features



Neural Message Passing

- At every iteration, every node receives features from its neighboring nodes.
- These features are then aggregated with an order invariant operation and combined with the current features with a learnable function



Neural Message Passing

- At every message passing step l, for every node do:

$$\underbrace{m_v^{(l+1)}}_{\text{Message}} = \underbrace{\sum_{u \in N_v}}_{\text{Aggregation overall all neighbors}} \underbrace{M^{(l)}}_{\text{Learnable function (e.g. MLP) with shared weights across the entire graph}}\left(h_u^{(l)}, h_v^{(l)}, h_{(u,v)}^{(0)}\right)$$

Gilmer et al. "Neural Message Passing for Quantum Chemistry". ICML 2017

Neural Message Passing

- At every message passing step l, for every node do:

$$m_v^{(l+1)} = \sum_{u \in N_v} M^{(l)}(h_u^{(l)}, h_v^{(l)}, h_{(u,v)}^{(0)})$$

$$h_v^{(l+1)} = \underbrace{U^{(l)}}_{\text{Learnable function (e.g. MLP) with shared weights across the entire graph}}(h_v^{(l)}, m_v^{(l+1)})$$

Embedding update

Learnable function (e.g. MLP) with shared weights across the entire graph

Gilmer et al. "Neural Message Passing for Quantum Chemistry". ICML 2017

Neural Message Passing

- At every message passing step l , for every node do:

$$m_v^{(l+1)} = \sum_{u \in N_v} M^{(l)}(h_u^{(l)}, h_v^{(l)}, h_{(u,v)}^{(0)})$$

$$h_v^{(l+1)} = U^{(l)}(h_v^{(l)}, m_v^{(l+1)})$$

Most Graph Neural Network Models can be seen as specific example of this formulation

Gilmer et al. "Neural Message Passing for Quantum Chemistry". ICML 2017

Neural Message Passing: An Example

$$m_v^{(l+1)} = \underbrace{\sum_{u \in N_v} \frac{h_u^{(l)}}{|N_v|}}$$

Average neighbors' feature embeddings

Neural Message Passing: An Example

New message

Previous embedding

$$h_v^{(l+1)} = \sigma\left(W^{(l+1)} m_v^{(l+1)} + B^{(l+1)} h_v^{(l)}\right)$$

Non-linearity

Learnable matrices, shared for all nodes

Combine node features with its neighbors'

Neural Message Passing: An Example

- We can use MLPs or even recurrent networks, instead of linear functions
- These are THE SAME for ALL nodes and edges!

$$h_v^{(l+1)} = \sigma\left(\underbrace{MLP_1^{(l+1)}}\, m_v^{(l+1)} + \underbrace{MLP_2^{(l+1)}}\, h_v^{(l)}\right)$$

Graph Convolutional Networks

$$m_v^{(l+1)} = \sum_{u \in N_v \cup \underbrace{\{v\}}_{\text{Self loop}}} \underbrace{\frac{h_u^{(l)}}{\sqrt{|N_v||N_u|}}}_{\text{Per neighbor degree normalization}}$$

Kipf and Welling. "Semi-Supervised Classification with Graph Convolutional Networks". ICLR 2016.

Graph Convolutional Networks

$$m_v^{(l+1)} = \sum_{u \in N_v \cup \{v\}} \frac{h_u^{(l)}}{\sqrt{|N_v||N_u|}}$$

$$h_v^{(l+1)} = \sigma\left(W^{(l+1)} m_v^{(l+1)}\right)$$

Same learnable matrix for self-loops and regular neighbors

Kipf and Welling. "Semi-Supervised Classification with Graph Convolutional Networks". ICLR 2016.

Graph Convolutional Networks

$$m_v^{(l+1)} = \sum_{u \in N_v \cup \{v\}} \frac{h_u^{(l)}}{\sqrt{|N_v||N_u|}}$$

$$h_v^{(l+1)} = \sigma\left(W^{(l+1)} m_v^{(l+1)}\right)$$

Matrix of weights is of size = #channels out x #channels in

Kipf and Welling. "Semi-Supervised Classification with Graph Convolutional Networks". ICLR 2016.

Graph Convolutional Networks

- We want to collect information from our neighbors and convert it to a new embedding

$$h_v^{(l+1)} = \sigma\left(W^{(l+1)} m_v^{(l+1)}\right)$$

Matrix of weights is of size = #channels out x #channels in

Graph Convolutional Networks

$$m_v^{(l+1)} = \sum_{u \in N_v \cup \{v\}} \frac{h_u^{(l)}}{\sqrt{|N_v||N_u|}}$$

- Unlike a normal image convolutional filter, here the neighbors are not regular (as they are in the image space), hence I have to do a permutation-invariant aggregation operation before the convolution.

Graph Convolutional Networks

$$m_v^{(l+1)} = \sum_{u \in N_v \cup \{v\}} \frac{h_u^{(l)}}{\sqrt{|N_v||N_u|}}$$

Aggregation

$$h_v^{(l+1)} = \sigma\left(W^{(l+1)} m_v^{(l+1)}\right)$$

Convolution

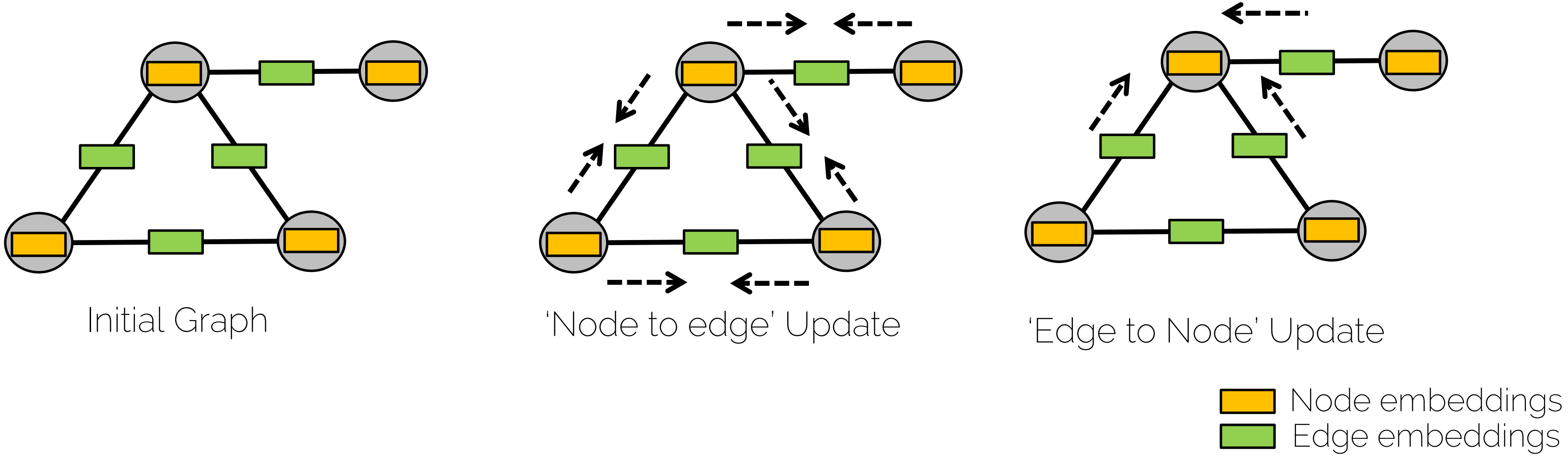
Kipf and Welling. "Semi-Supervised Classification with Graph Convolutional Networks". ICLR 2016.

What About Edge Embeddings?

- The framework we've presented is only suited to learn node embeddings. But what happens if our focus is on edge features?
- At least, two options:
  - Work on the 'dual' or 'line' graph
    - E.g. Chen et al. "Supervised Community Detection with Line Graph Neural Networks", ICLR 2019.
  - Use a more general formulation that admits edge updates
    - E.g. Battaglia et al. "Relational inductive biases, deep learning, and graph networks". arXiv 2018

A More General Framework

- We can divide the propagation process in two steps: ‘node to edge’ and ‘edge to node’ updates.



Battaglia et al. “Relational inductive biases, deep learning, and graph networks”. 2018

‘Node to edge’ updates

- At every message passing step l, first do:

$$h_{(i,j)}^{(l)} = \mathcal{N}_e \left([h_i^{(l-1)}, h_{(i,j)}^{(l-1)}, h_j^{(l-1)}] \right)$$

Embedding of node i in the precious message passing step

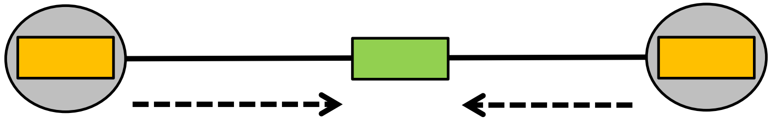
Embedding of edge (i,j) in the previous message passing step

Embedding of node j in the precious message passing step

'Node to edge' updates

- At every message passing step l, first do:

$$h_{(i,j)}^{(l)} = \mathcal{N}_e\left([h_i^{(l-1)}, h_{(i,j)}^{(l-1)}, h_j^{(l-1)}]\right)$$

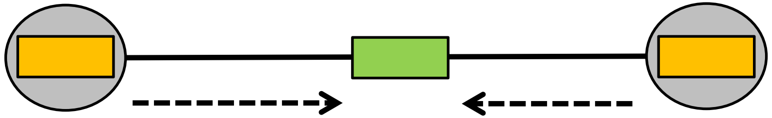


‘Node to edge’ updates

- At every message passing step l, first do:

$$h_{(i,j)}^{(l)} = \mathcal{N}_e\left(\left[h_i^{(l-1)}, h_{(i,j)}^{(l-1)}, h_j^{(l-1)}\right]\right)$$

Learnable function (e.g. MLP) with shared weights across the entire graph



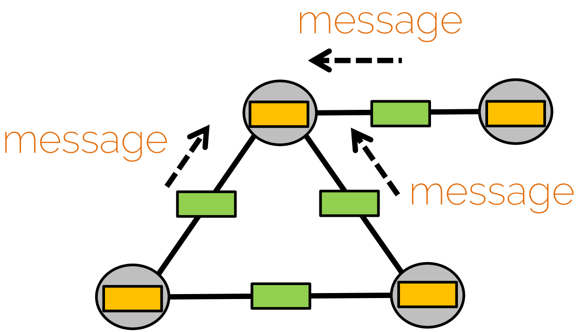
‘Edge to node’ updates

- After a round of edge updates, each edge embedding contains information about its pair of incident nodes
- Then, edge embeddings are used to update nodes:

$$m_i^{(l)} = \Phi\left(\left\{h_{(i,j)}^{(l)}\right\}_{j \in N_i}\right)$$

Order invariant operation (e.g. sum, mean, max)

Neighbors of node i



'Edge to node' updates

- After a round of edge updates, each edge embedding contains information about its pair of incident nodes
- Then, edge embeddings are used to update nodes:

$$m_i^{(l)} = \Phi\left(\left\{h_{(i,j)}^{(l)}\right\}_{j \in N_i}\right)$$

$$h_i^{(l)} = \underbrace{\mathcal{N}_v}\left(\left[m_i^{(l)}, h_i^{(l-1)}\right]\right)$$

Learnable function (e.g. MLP) with shared weights across the entire graph

The aggregation provides each node embedding with contextual information about its neighbors

Remarks

- **Main goal**: obtaining node and edge embeddings that contain *context information* encoding graph topology and neighbor's feature information.
- After repeating the node and edge updates for l steps, each node (resp. edge) embedding contains information about all nodes (resp. edge) at distance l (resp. l – 1) → Think of iterations as layers in a CNN
- Observe that all operations used are differentiable, hence, MPNs can be used within end-to-end pipelines
- There is vast literature on different instantiations, as well as variations of the MPN framework we presented. See Battaglia et al. for an extensive review.

Message Passing Networks for Computer Vision

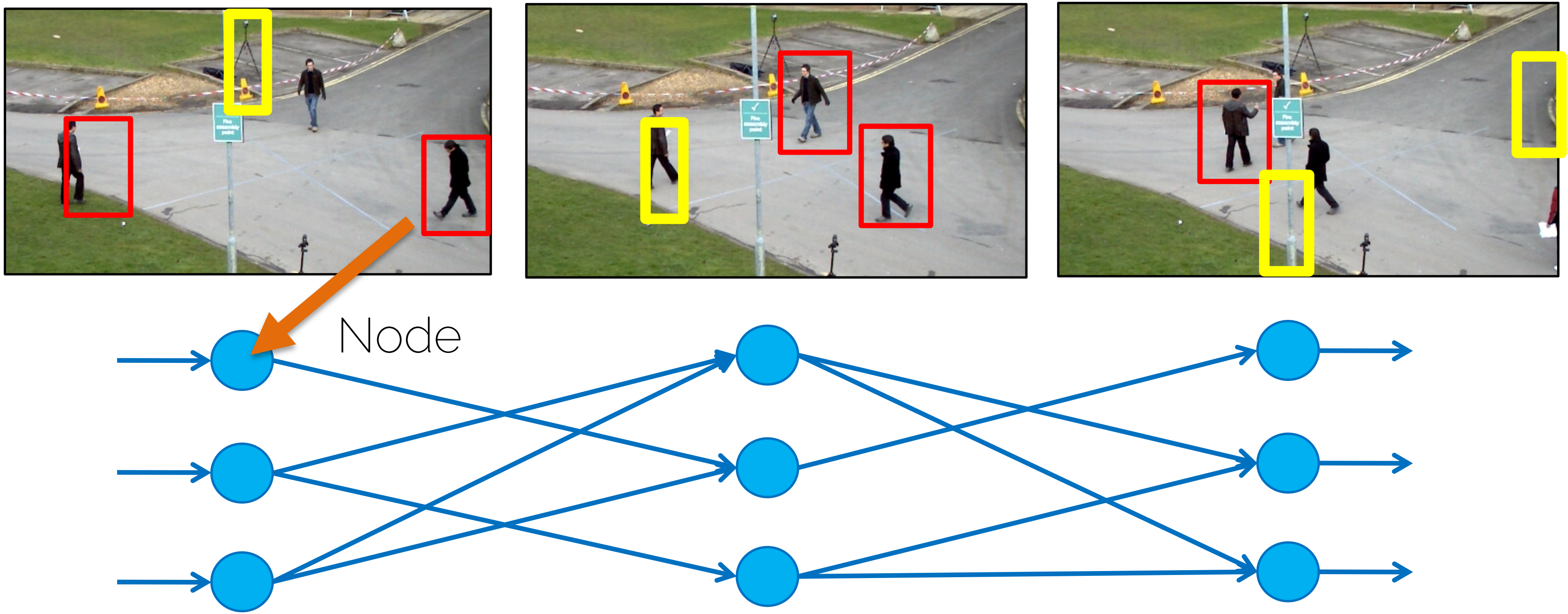
Different challenges

- Multiple objects of the same type
- Heavy occlusions
- Appearance is often very similar

Multi-object tracking with graphs

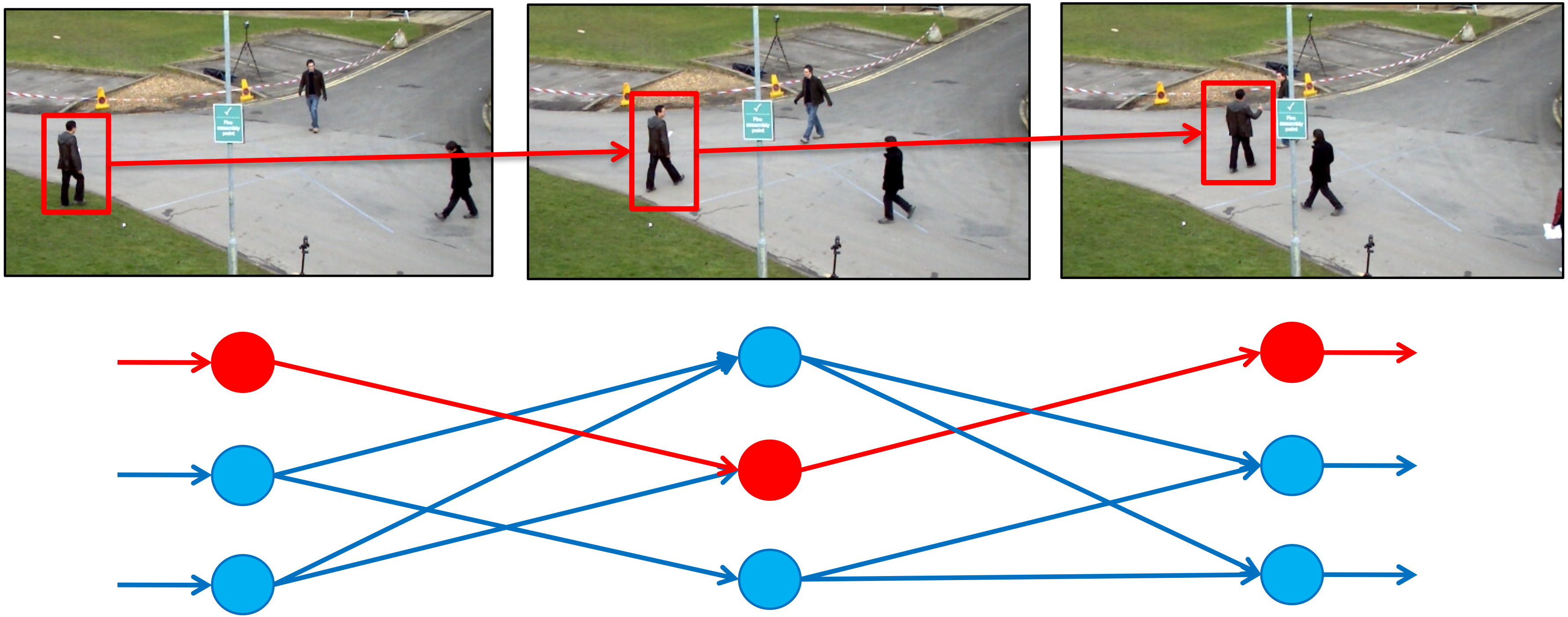
Step 1: Object detection

Multi-object tracking with graphs



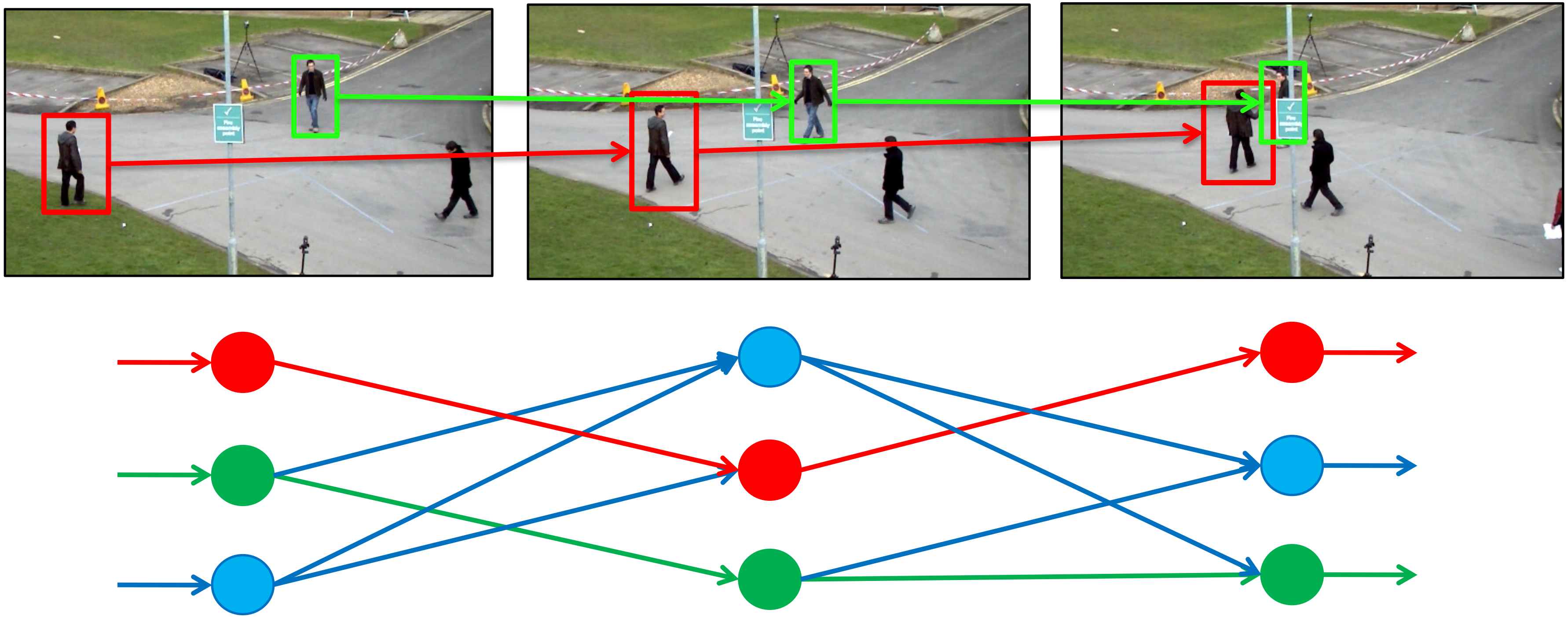
Graphical model

Multi-object tracking with graphs



L. Leal-Taixé et al. "Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker." ICCVW2011

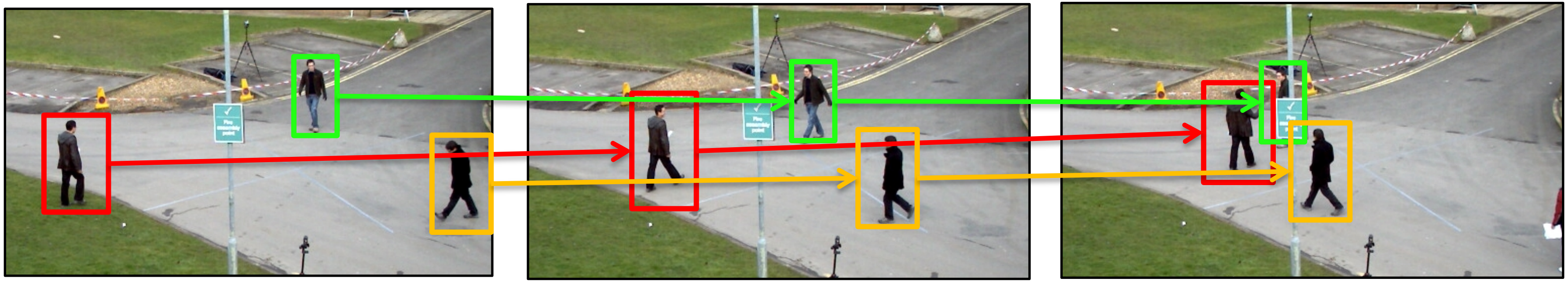
Multi-object tracking with graphs



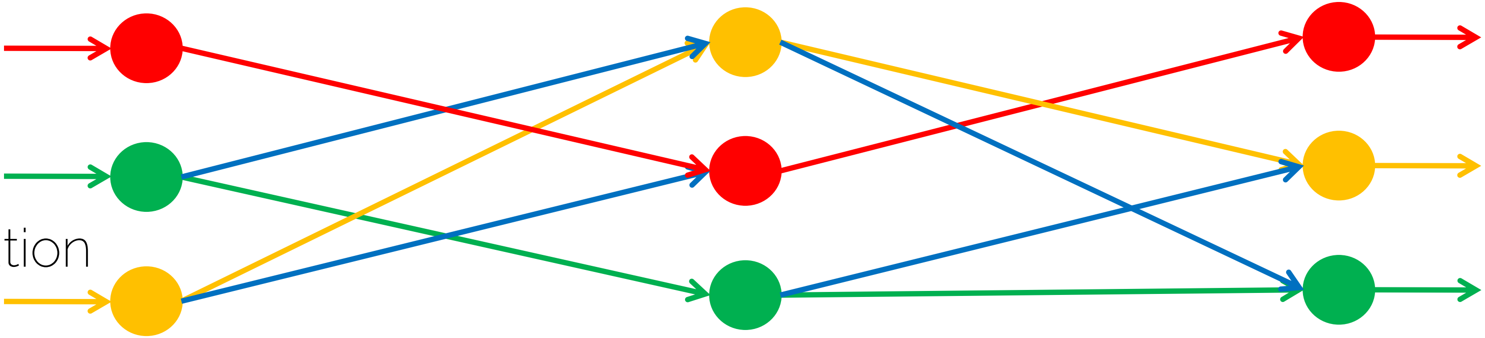
L. Leal-Taixé et al. "Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker." ICCVW2011

Multi-object tracking with graphs

Step 1: Object detection

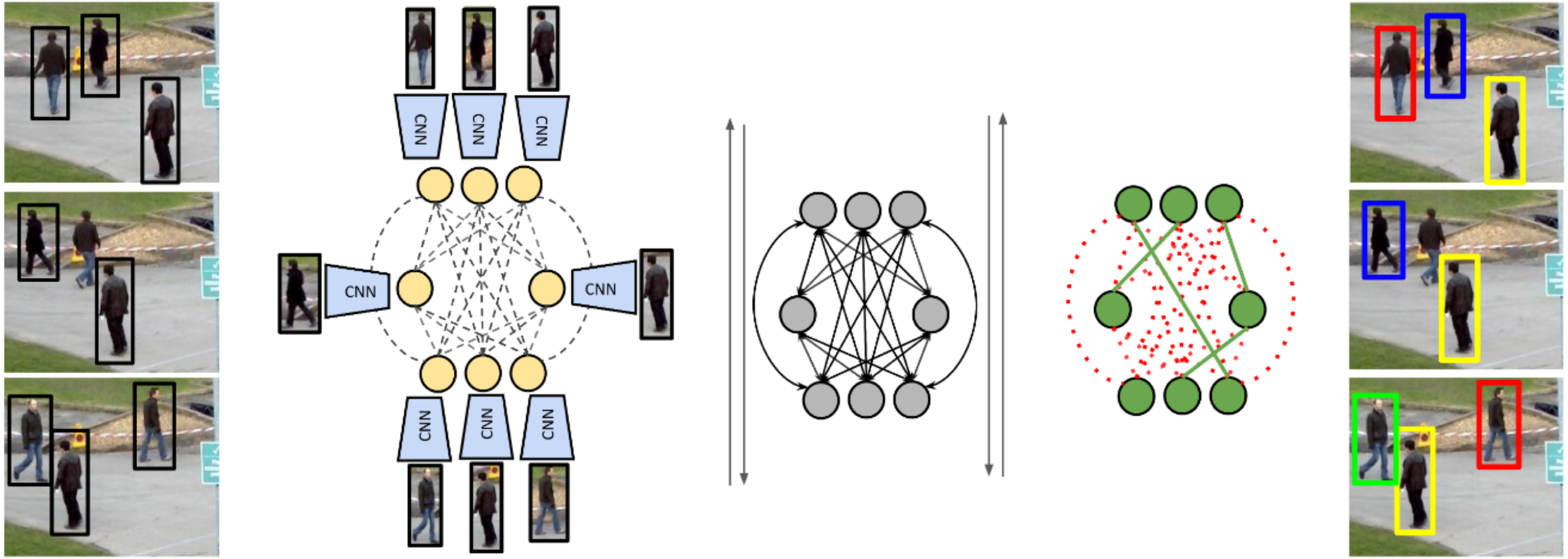


Step 2: Data association



L. Leal-Taixé et al. "Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker." ICCVW2011

MOT with MPN: Overview

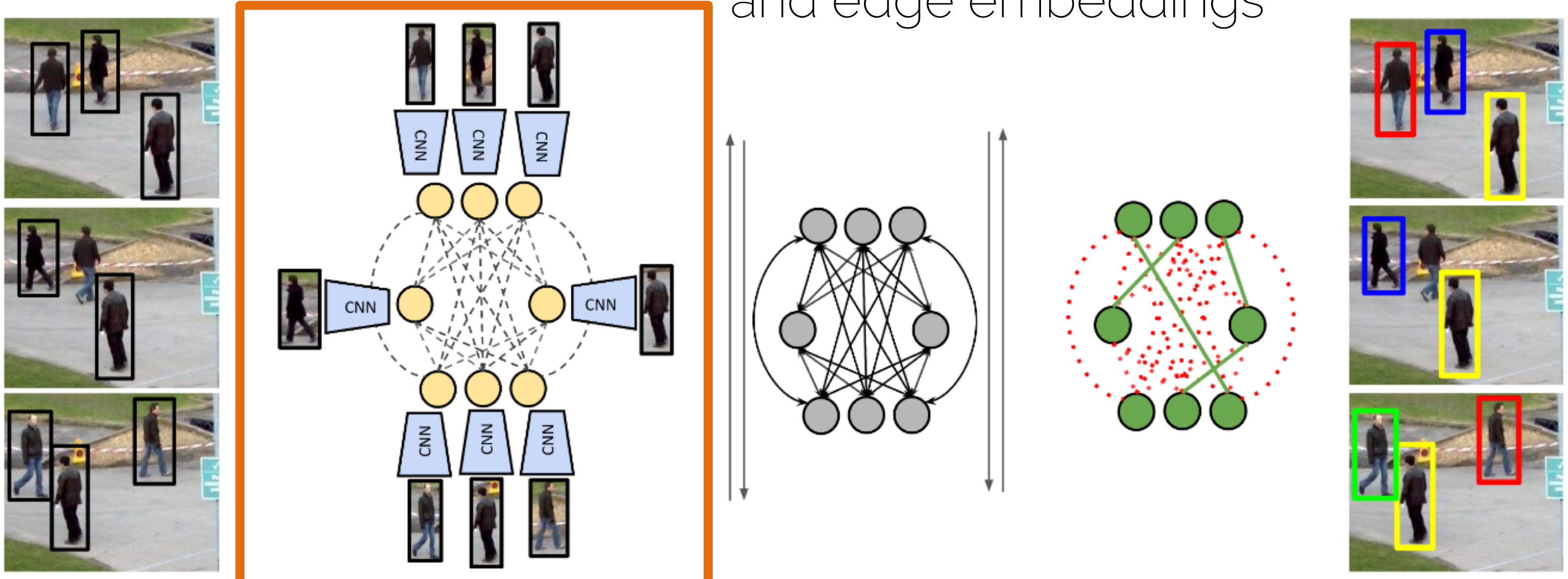


(a) Input (b) Graph Construction + Feature Encoding (c) Neural Message Passing (d) Edge Classification (e) Output

G. Brasó and L. Leal-Taixé. "Learning a Neural Solver for Multiple Object Tracking", CVPR 2020

MOT with MPN: Overview

Encode appearance and scene geometry cues into node and edge embeddings

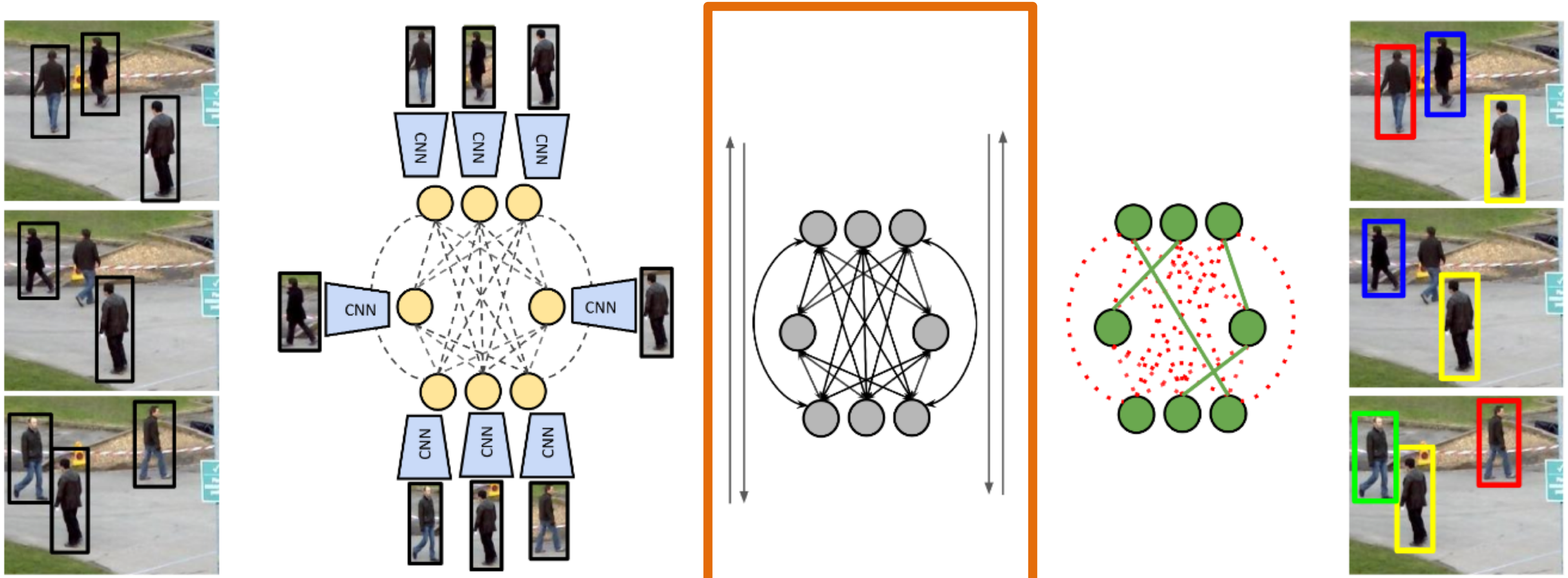


(a) Input (b) Graph Construction + Feature Encoding (c) Neural Message Passing (d) Edge Classification (e) Output

G. Brasó and L. Leal-Taixé. "Learning a Neural Solver for Multiple Object Tracking", CVPR 2020

MOT with MPN: Overview

Propagate cues across the entire graph with neural message passing

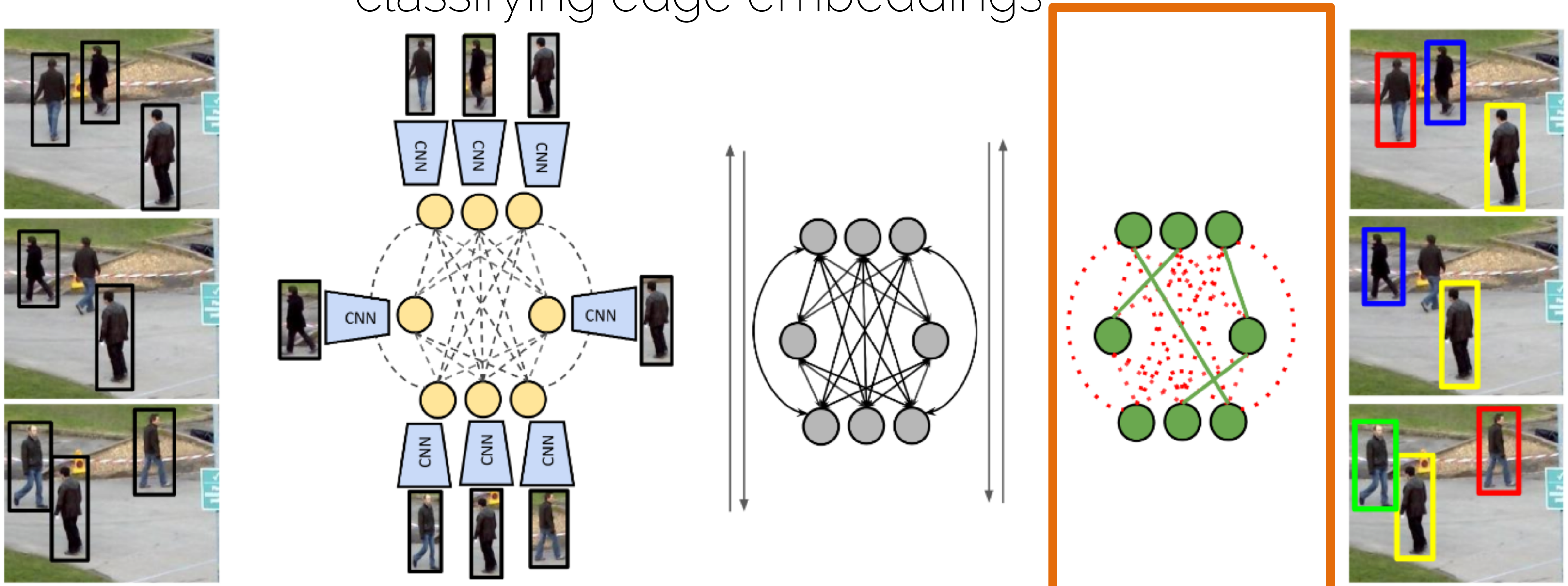


(a) Input (b) Graph Construction + Feature Encoding (c) Neural Message Passing (d) Edge Classification (e) Output

G. Brasó and L. Leal-Taixé. "Learning a Neural Solver for Multiple Object Tracking", CVPR 2020

MOT with MPN: Overview

Learn to directly predict solutions of the tracking graph problem by classifying edge embeddings

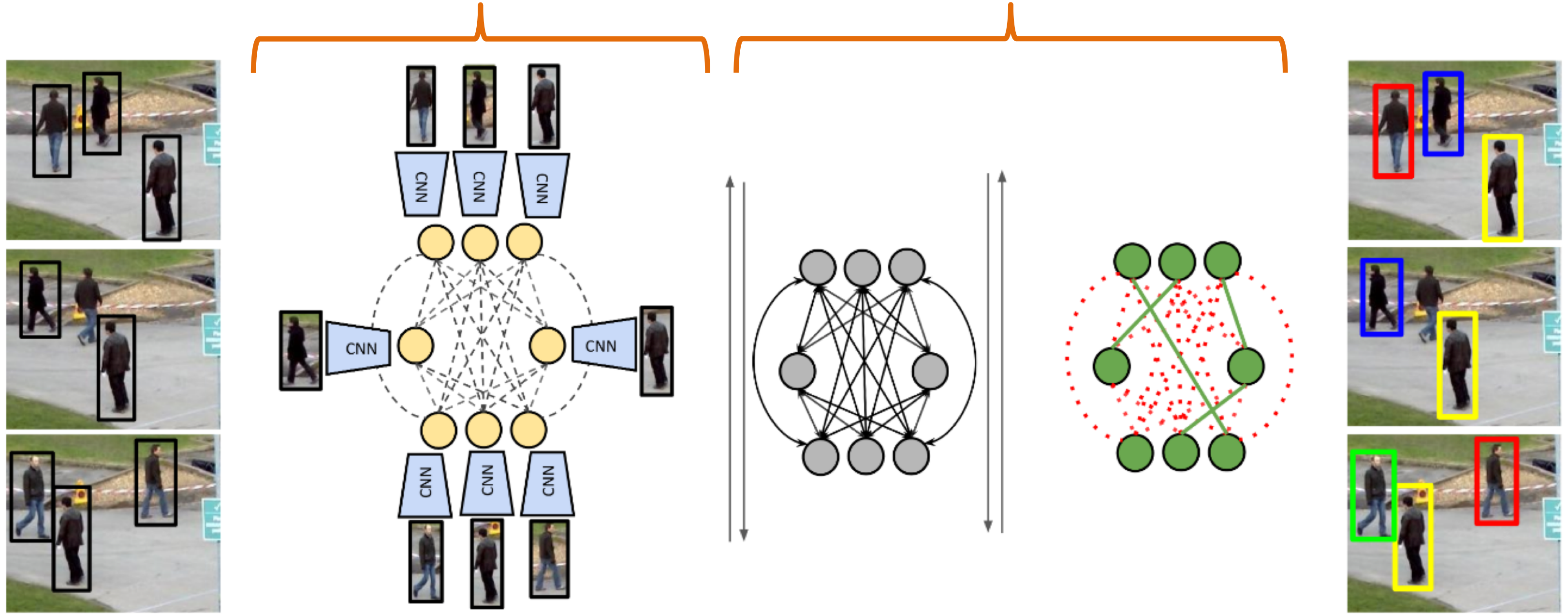


(a) Input (b) Graph Construction + Feature Encoding (c) Neural Message Passing (d) Edge Classification (e) Output

G. Brasó and L. Leal-Taixé. "Learning a Neural Solver for Multiple Object Tracking", CVPR 2020

MOT with MPN: Overview

Feature Extraction   Learnable Data Association

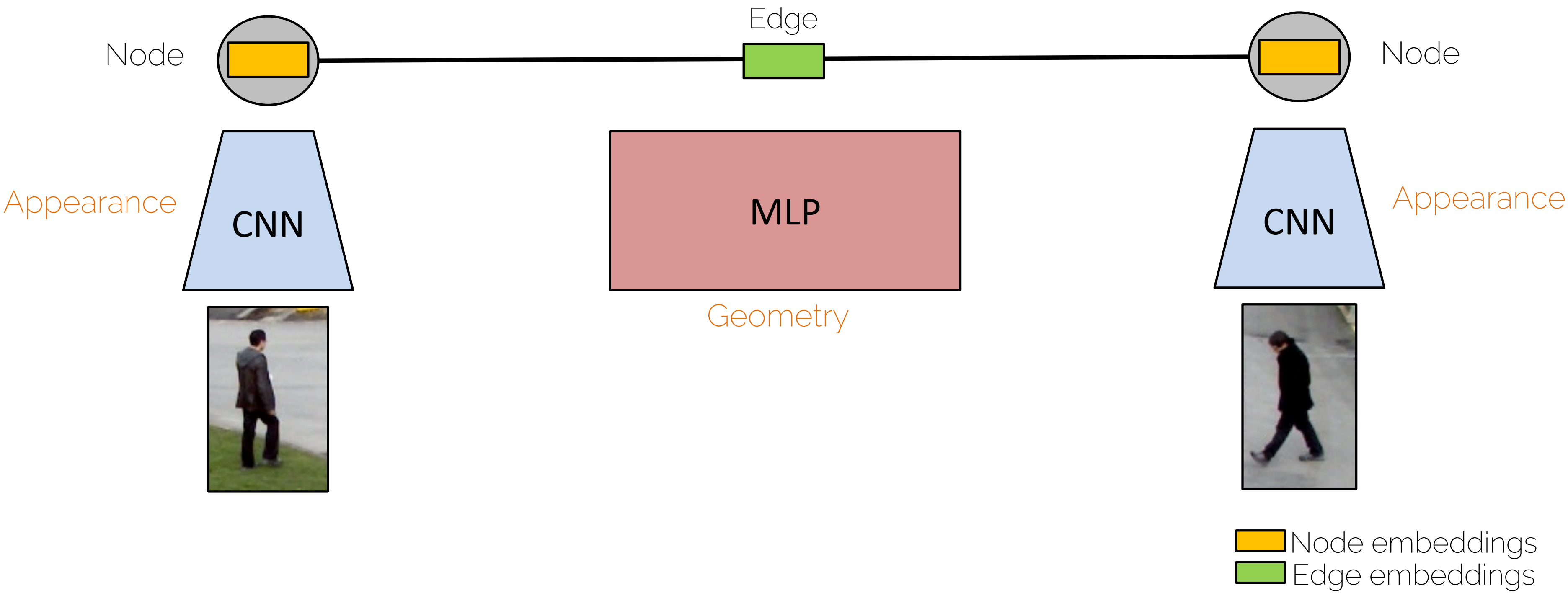


(a) Input (b) Graph Construction + Feature Encoding (c) Neural Message Passing (d) Edge Classification (e) Output

G. Brasó and L. Leal-Taixé. "Learning a Neural Solver for Multiple Object Tracking", CVPR 2020

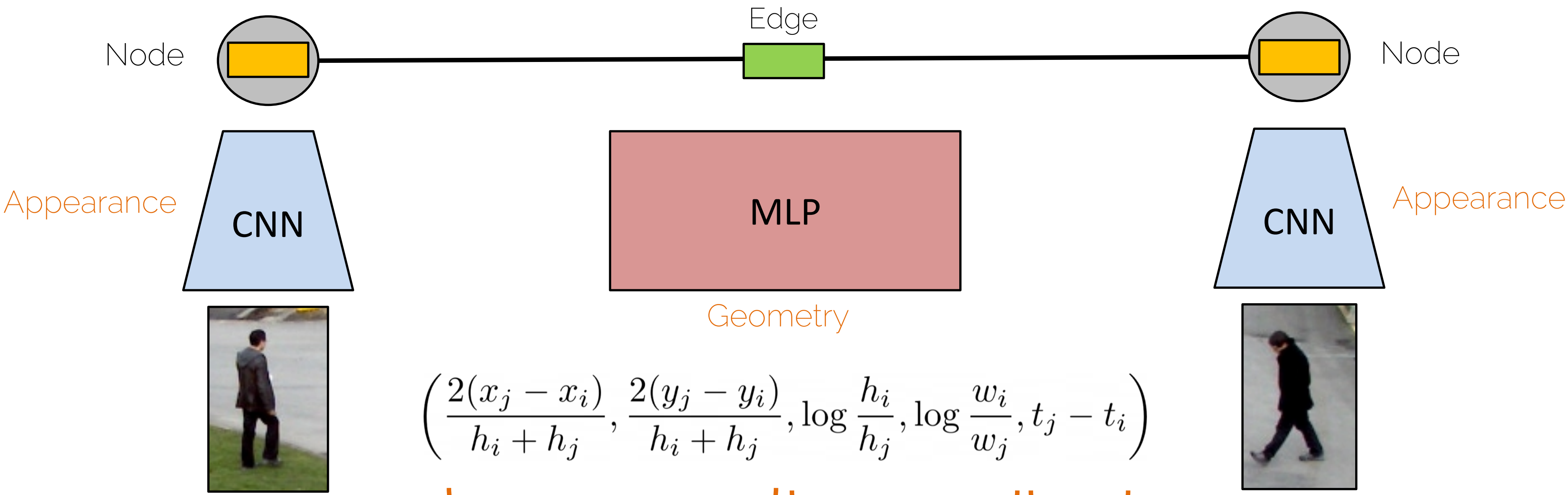
Feature encoding

- Appearance and geometry encodings



Feature encoding

- Appearance and geometry encodings

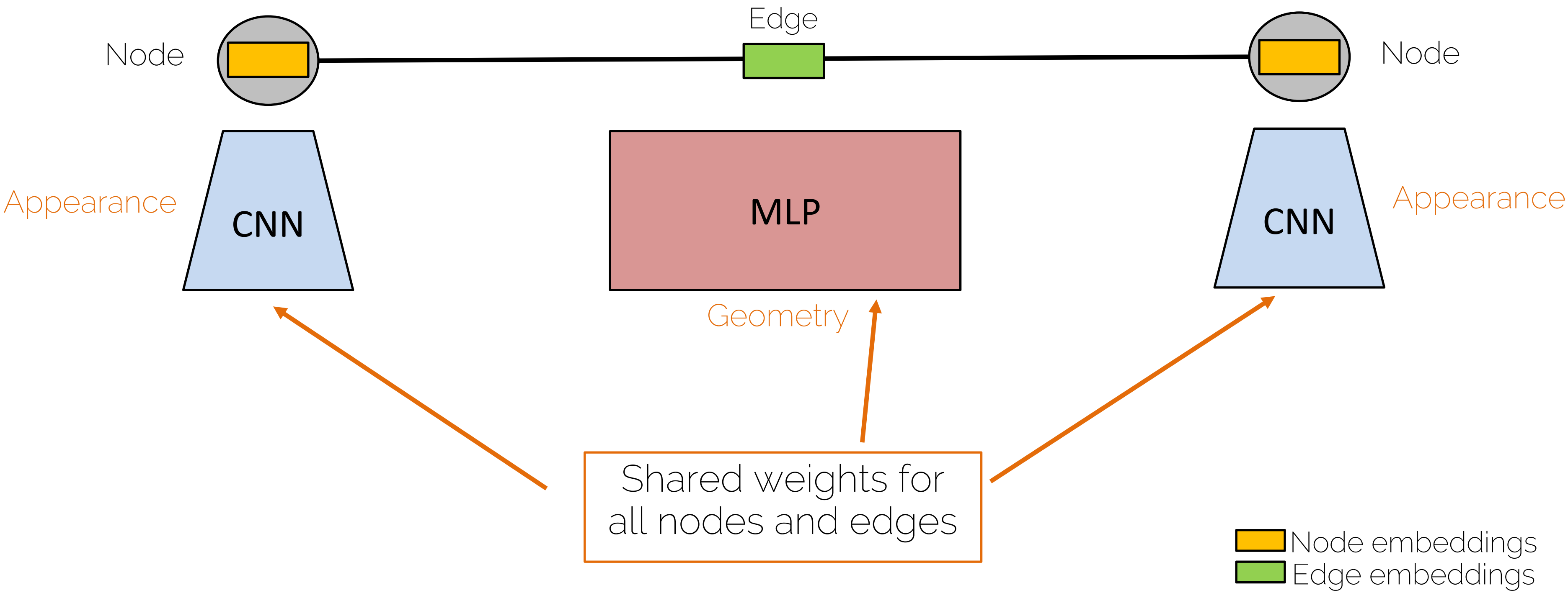


$$\Bigg(\underbrace{\frac{2(x_j - x_i)}{h_i + h_j}, \frac{2(y_j - y_i)}{h_i + h_j}}_{\text{Relative Box Position}}, \underbrace{\log \frac{h_i}{h_j}, \log \frac{w_i}{w_j}}_{\text{Relative Box Size}}, \underbrace{t_j - t_i}_{\text{Time Difference}}\Bigg)$$

Node embeddings
Edge embeddings

Feature encoding

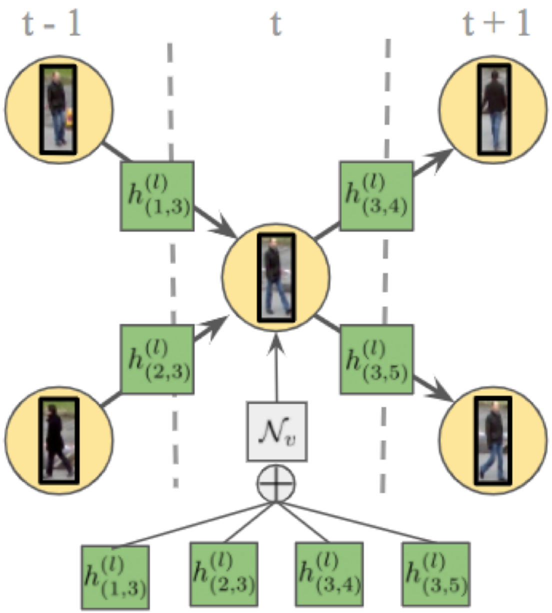
- Appearance and geometry encodings



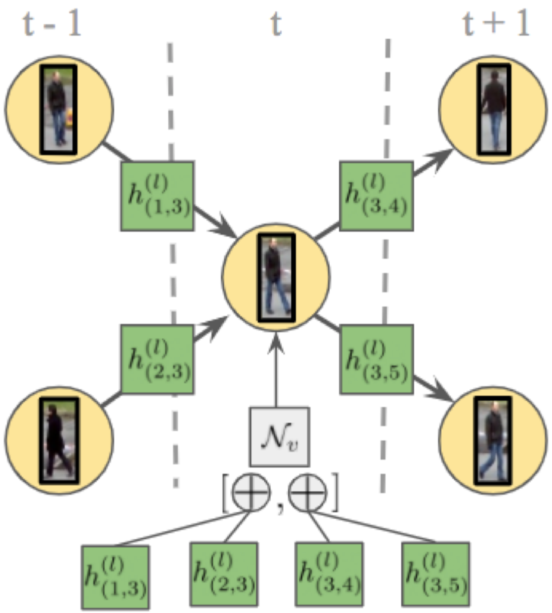
Feature encoding

- Goal: propagate these embeddings across the entire graph in order to obtain new embeddings encoding high-order information among detections

Time-aware Message Passing



All edge embeddings are aggregated at once



Aggregation of edge embeddings is separated between past / future frames

Classifying edges

- After several iterations of message passing, each edge embedding contains high-order information about other detections
- We feed the embeddings to an MLP that predicts whether an edge is active/inactive

$$\mathcal{L} = \frac{-1}{|E|} \underbrace{\sum_{l=l_0}^{l=L}}_{\text{Sum over the last steps}} \sum_{(i,j)\in E} \underbrace{w}_{\text{Weight to balance active / inactive edges}} \cdot \underbrace{y_{(i,j)} \log(\boxed{\hat{y}^{(l)}_{(i,j)}}) + (1 - y_{(i,j)}) \log(1 - \boxed{\hat{y}^{(l)}_{(i,j)}})}_{\text{Binary cross-entropy}}$$

Edge predictions (w. sigmoid) at iteration l

Obtaining final solutions

- After classifying edges, we get a prediction between 0 and 1 for each edge in the graph.
- We use a simple rounding scheme to obtain the final edge values 0/1 that map to trajectories

- The overall method is reasonably fast (~6 fps) and achieves SOTA in the MOT Challenge by a significant margin

Video object segmentation

- Goal: Generate accurate and temporally consistent pixel masks for objects in a video sequence.

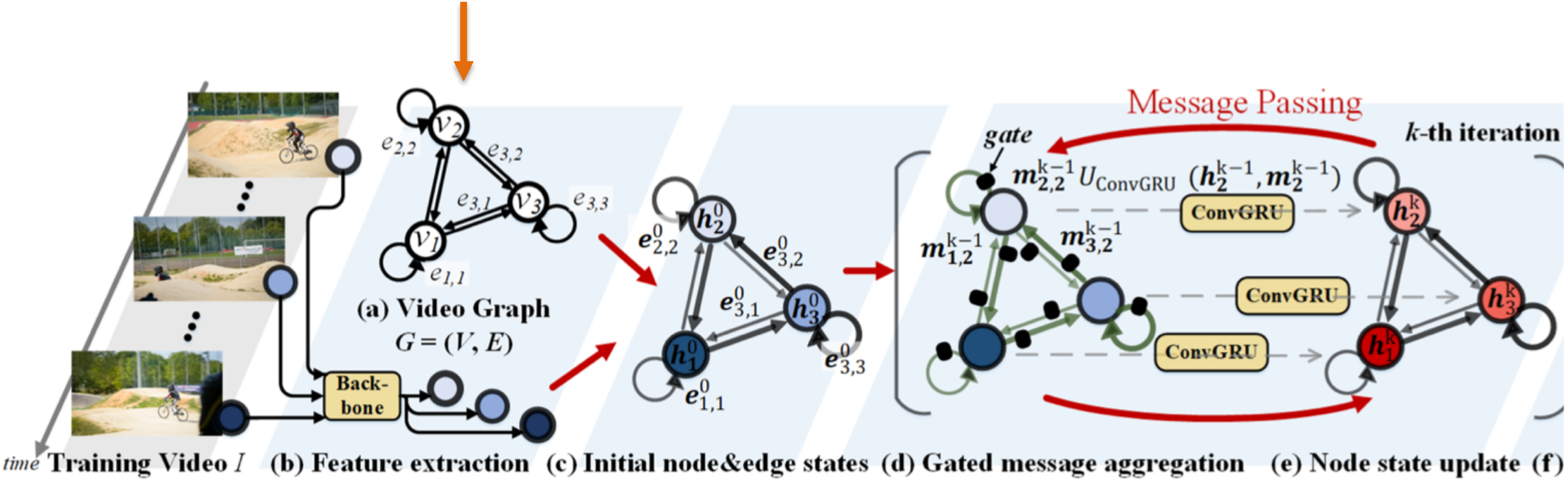
Video object segmentation

- Main idea: Model the temporal consistency through a Graph Neural Network.
- Each node is a frame, and information is passed among frames to obtain a consistent mask as output

W. Wang et al. „Zero-Shot Video Object Segmentation via Attentive Graph Neural Networks". ICCV 2019.

Video object segmentation

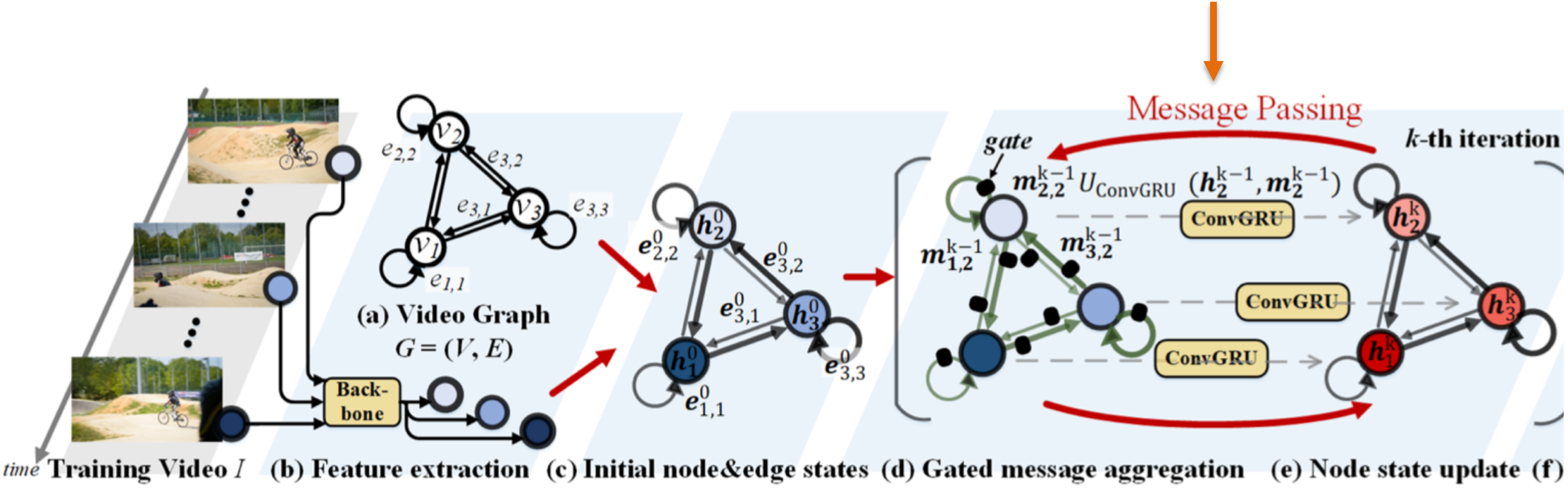
Features extraction with DeepLabV3
to construct the initial embeddings



W. Wang et al. „Zero-Shot Video Object Segmentation via Attentive Graph Neural Networks". ICCV 2019.

Video object segmentation

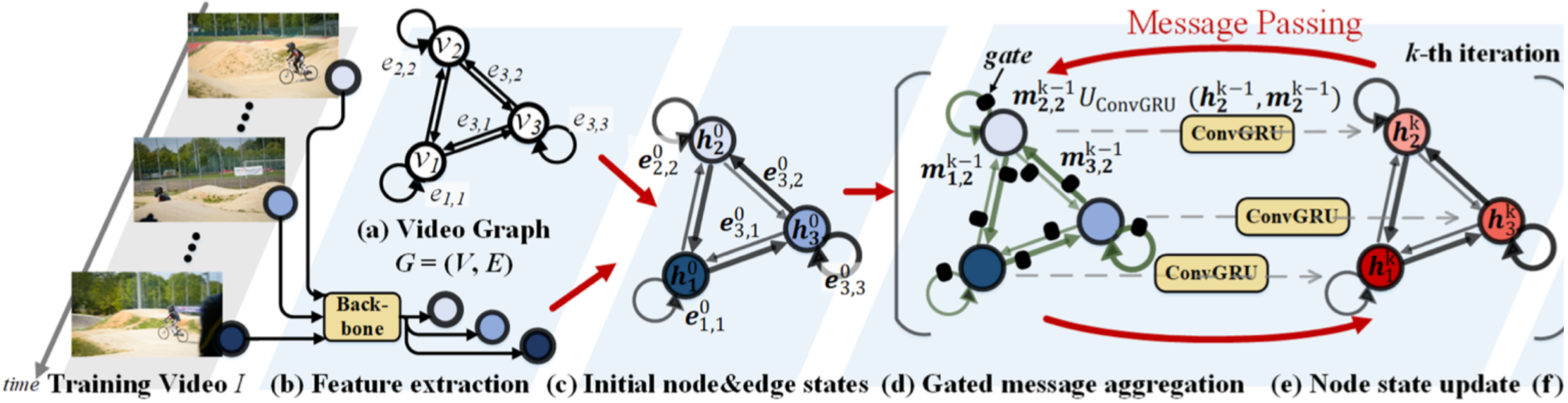
Message passing with convolutional recurrent networks, since we need to preserve the spatial information (we still want to get pixel outputs)



W. Wang et al. „Zero-Shot Video Object Segmentation via Attentive Graph Neural Networks". ICCV 2019.

Video object segmentation

- But each pixel is not equally important, so they further propose to use attention → what is that?

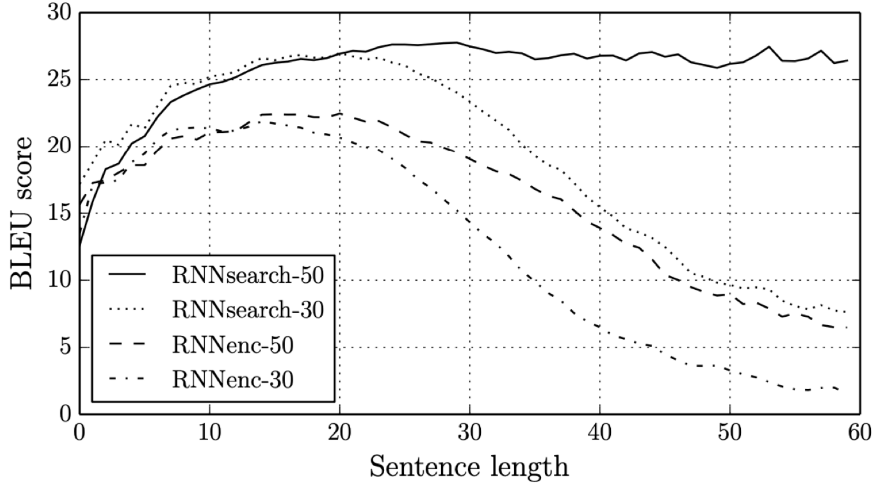


W. Wang et al. „Zero-Shot Video Object Segmentation via Attentive Graph Neural Networks". ICCV 2019.

Attention

The problem

- For very long sentences, the score for machine translation really goes down after 30-40 words.



With attention

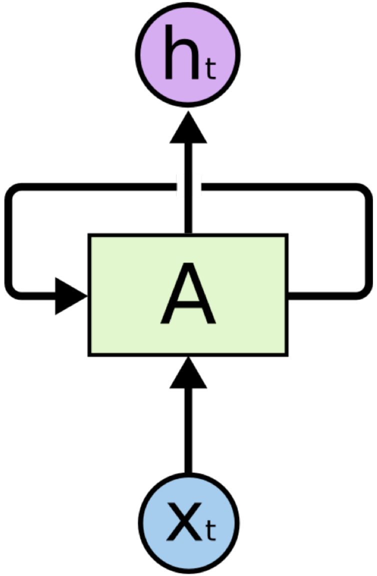
Performance degradation

Bahdanau et al 2014. Neural machine translation by jointly learning to align and translate.

Basic structure of a RNN

- We want to have notion of "time" or "sequence"

Hidden
state



$$\mathbf{A}_t = \boldsymbol{\theta}_c \mathbf{A}_{t-1} + \boldsymbol{\theta}_x \mathbf{x}_t$$

Previous
hidden
state

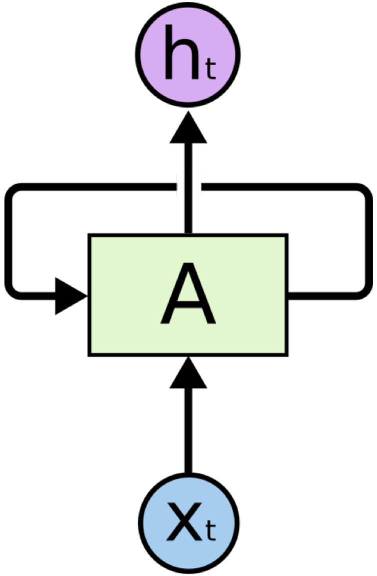
input

Image: Christopher Olah - Understanding LSTMs

Basic structure of a RNN

- We want to have notion of "time" or "sequence"

Hidden state

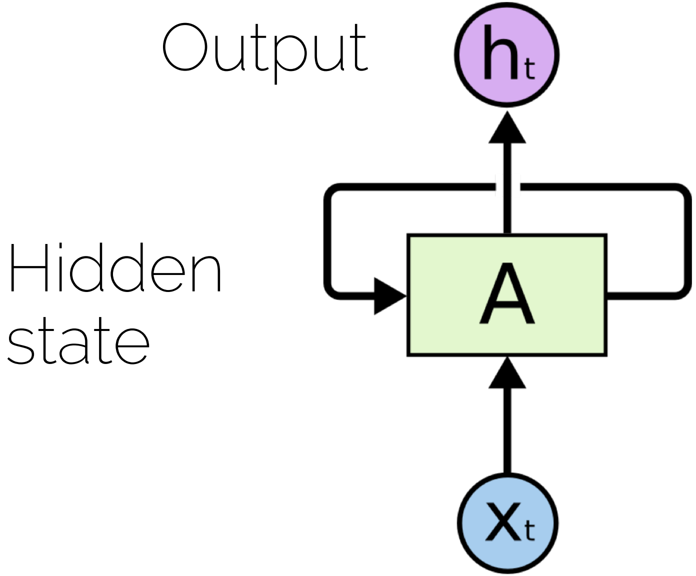


$$\mathbf{A}_t = \boldsymbol{\theta}_c \mathbf{A}_{t-1} + \boldsymbol{\theta}_x \mathbf{x}_t$$

Parameters to be learned

Basic structure of a RNN

- We want to have notion of "time" or "sequence"



$$\mathbf{A}_t = \boldsymbol{\theta}_c \mathbf{A}_{t-1} + \boldsymbol{\theta}_x \mathbf{x}_t$$

$$\mathbf{h}_t = \boldsymbol{\theta}_h \mathbf{A}_t$$

Same parameters for each time step = generalization!

Basic structure of a RNN

- Unrolling RNNs

Hidden state is the same

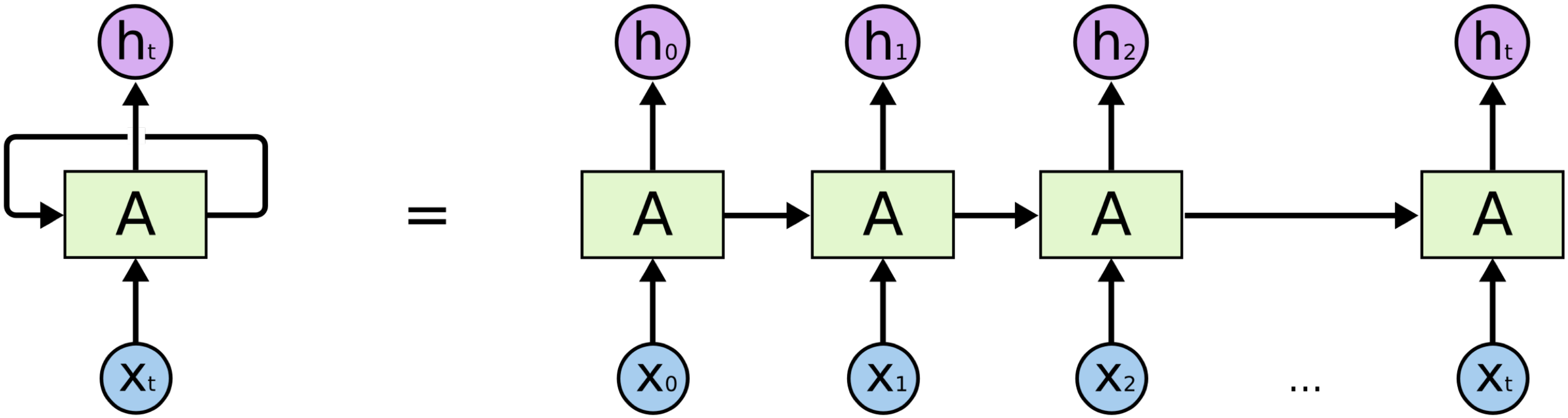
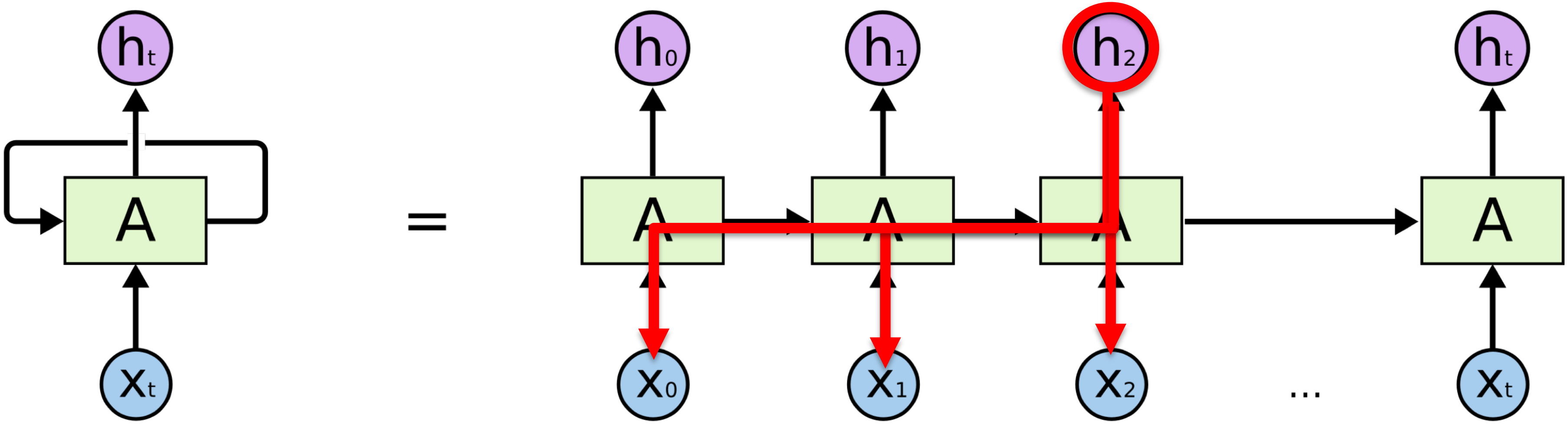


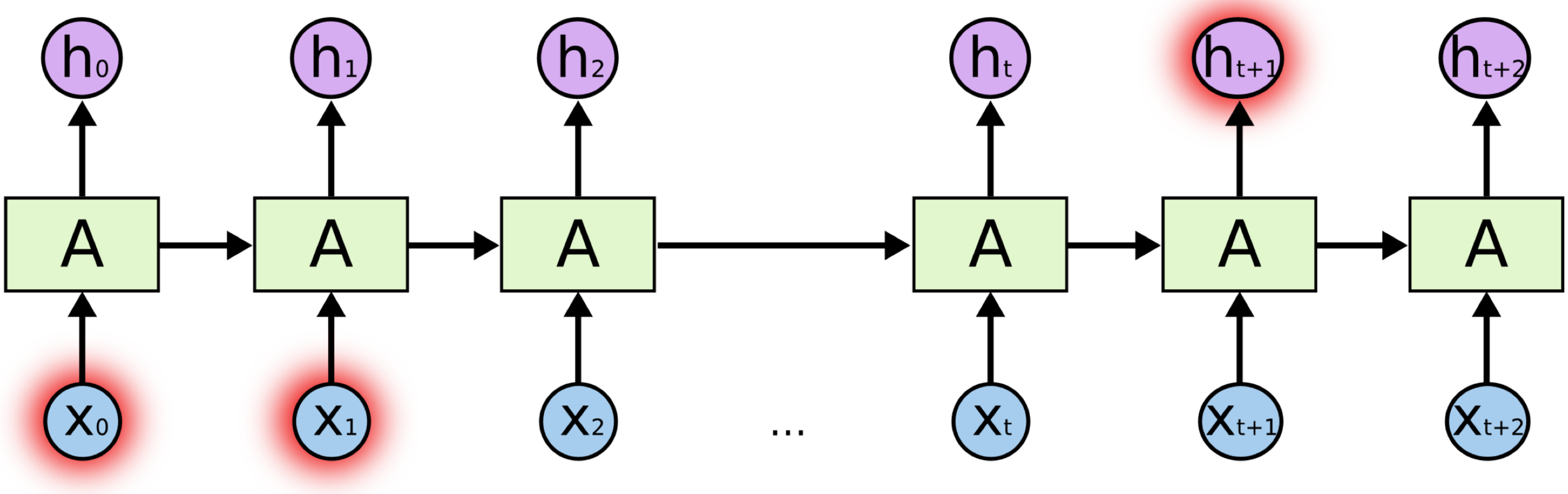
Image: Christopher Olah - Understanding LSTMs

Basic structure of a RNN

- Unrolling RNNs



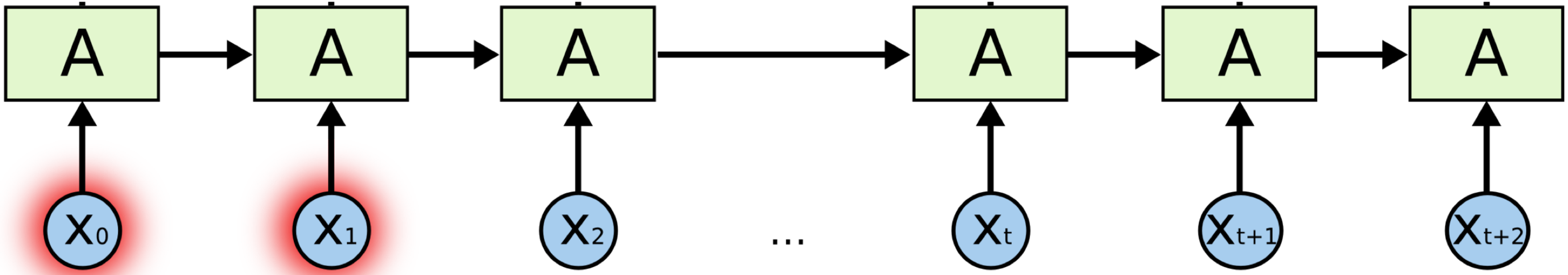
Long-term dependencies



I moved to Germany … so I speak German fluently

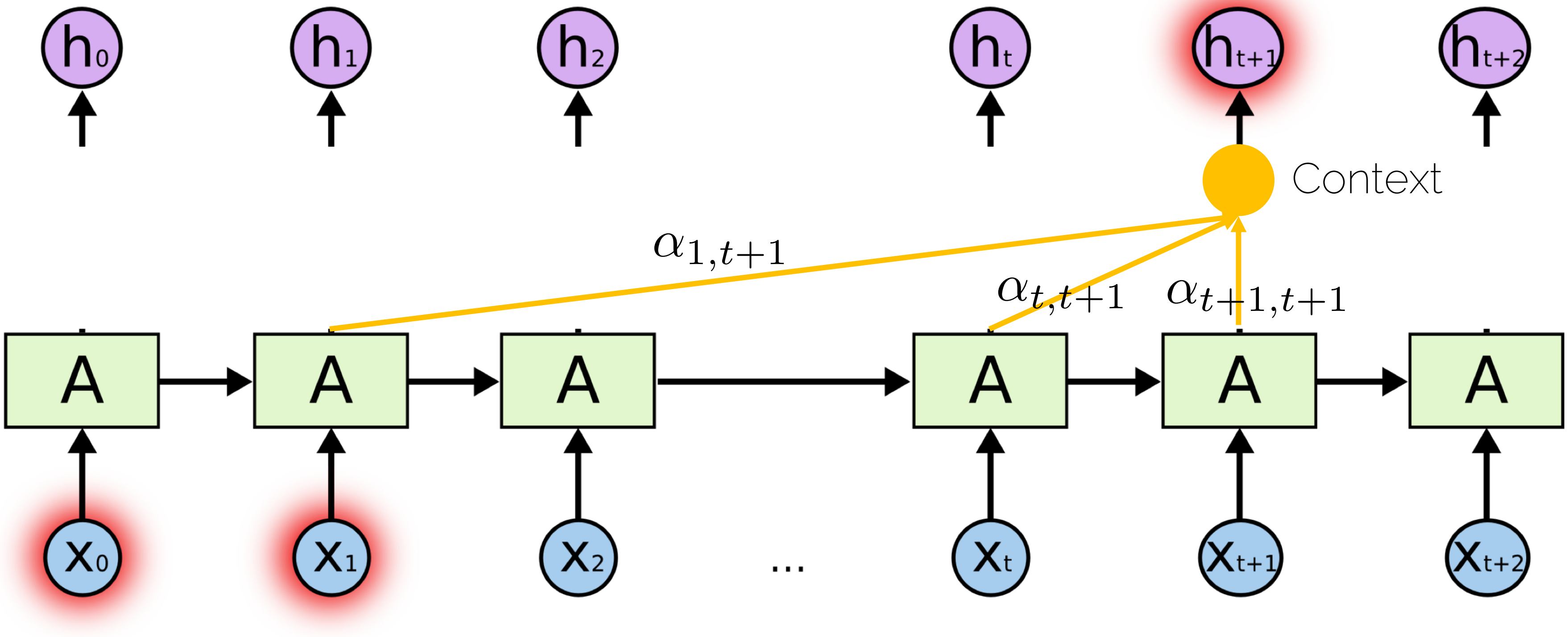
Attention: intuition

ATTENTION: Which hidden states are more important to predict my output?



I moved to Germany … so I speak German fluently

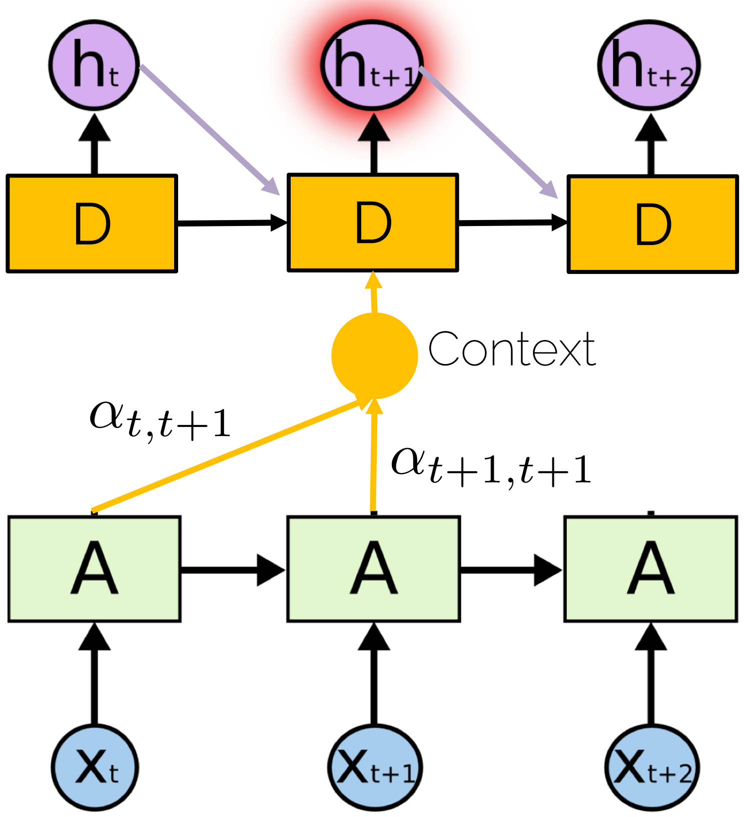
Attention: intuition



I moved to Germany … so I speak German fluently

Attention: architecture

- A decoder processes the information
- Decoders take as input:
  - Previous decoder hidden state
  - Previous output
  - Attention



Attention

- $\alpha_{1,t+1}$ indicates how much the word in the position 1 is important to translate the word in position $t+1$

- The context aggregates the attention

$$c_{t+1} = \sum_{k=1}^{t+1} \alpha_{k,t+1} a_k$$

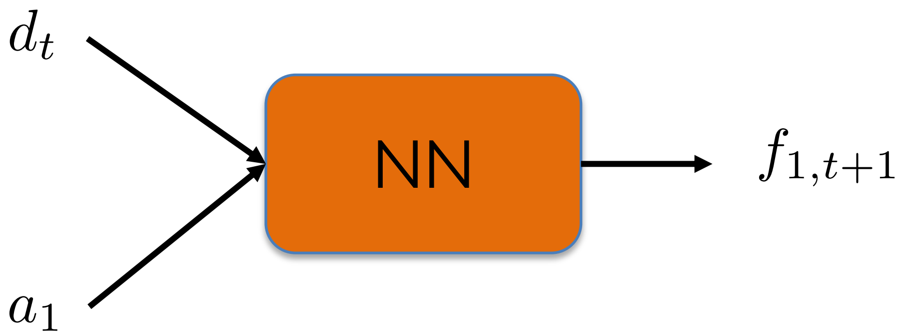
- Soft attention: All attention masks alpha sum up to 1

Computing the attention mask

- We can train a small neural network

Previous state of the decoder

Hidden state of the encoder

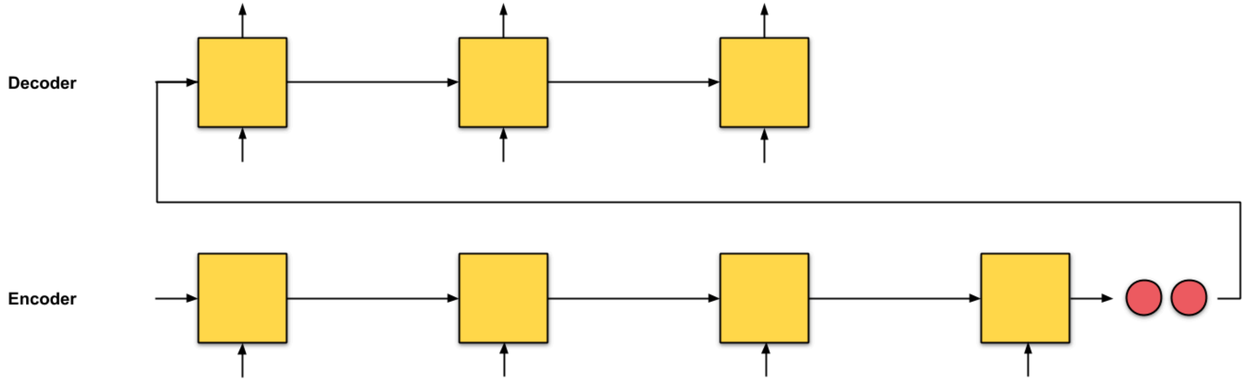


- Normalize

$$\alpha_{1,t+1} = \frac{\exp^{f_{1,t+1}}}{\sum_{k=1}^{t+1} \exp^{f_{k,t+1}}}$$

Seq2Seq

- How do we translate?
- First read *the whole* sentence in language 1.
- *Afterwards*, translate the whole sentence in language 2.



Sutskever et al. „Sequence to Sequence Learning with Neural Networks". NIPS 2014
Picture from: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

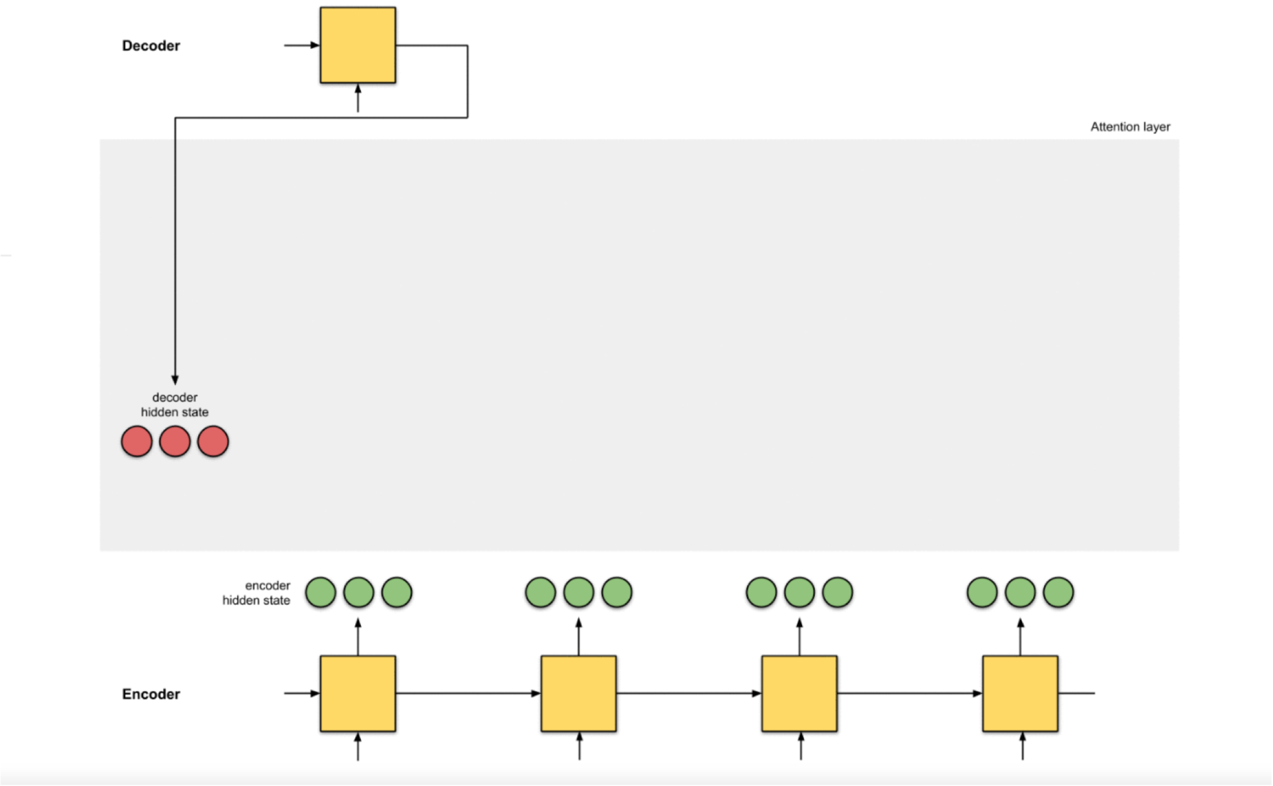
Seq2Seq + Attention?

- If the sentence is very long, we might have forgotten what was said at the beginning.

- Solution: take "notes" of keywords as we read the sentence in language 1.

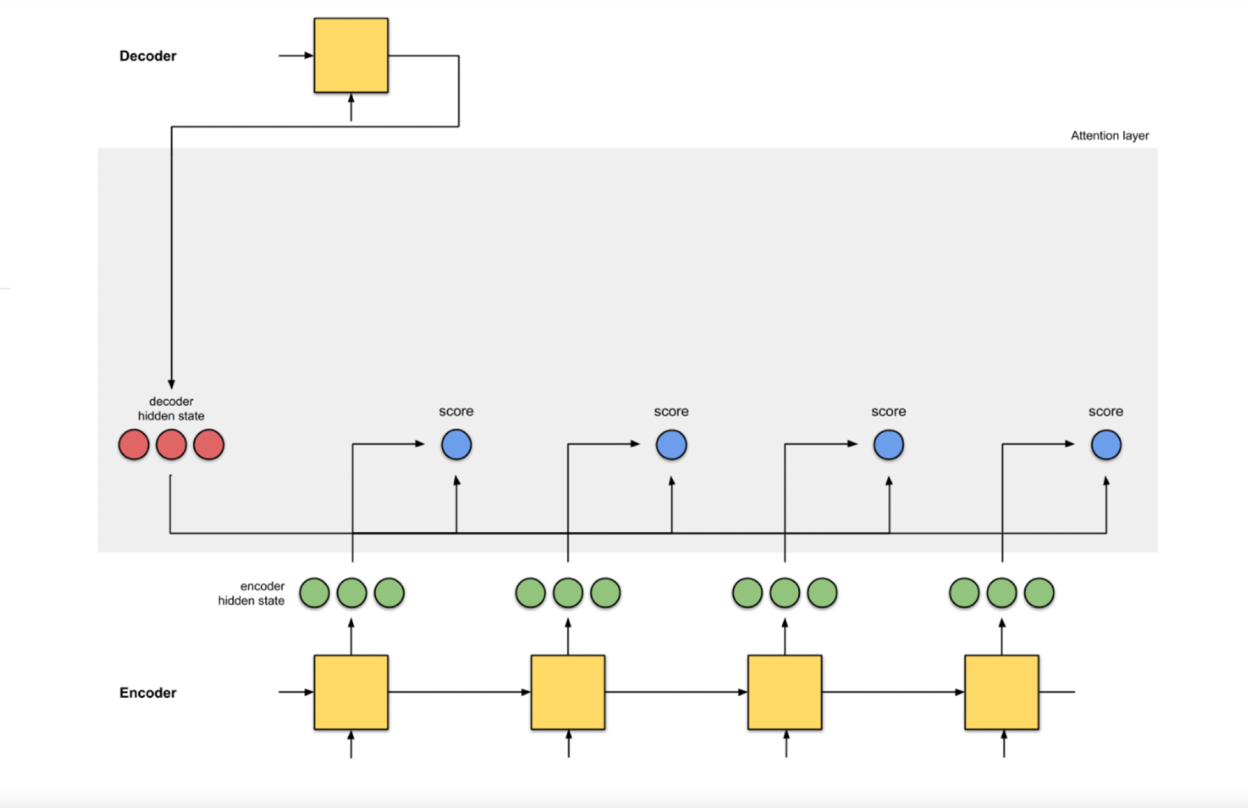
- Use attention!

Seq2Seq + Attention



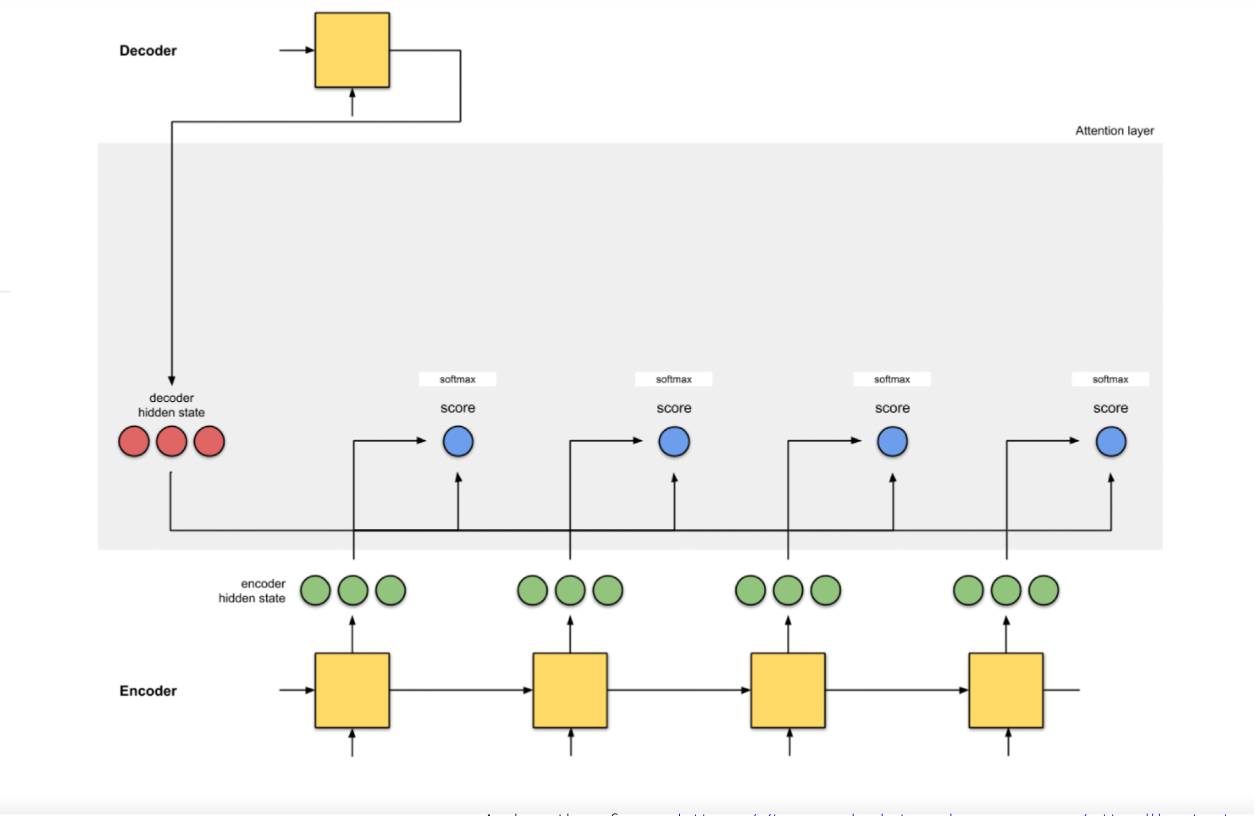
Animation from: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Seq2Seq + Attention



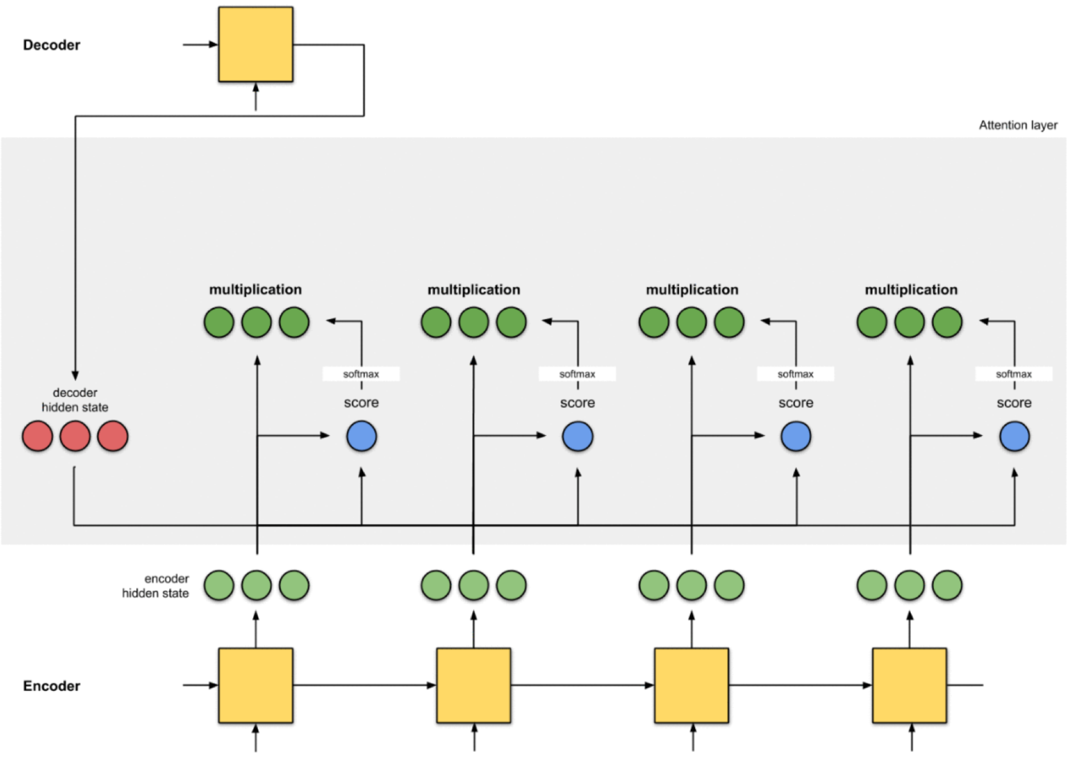
Animation from: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Seq2Seq + Attention



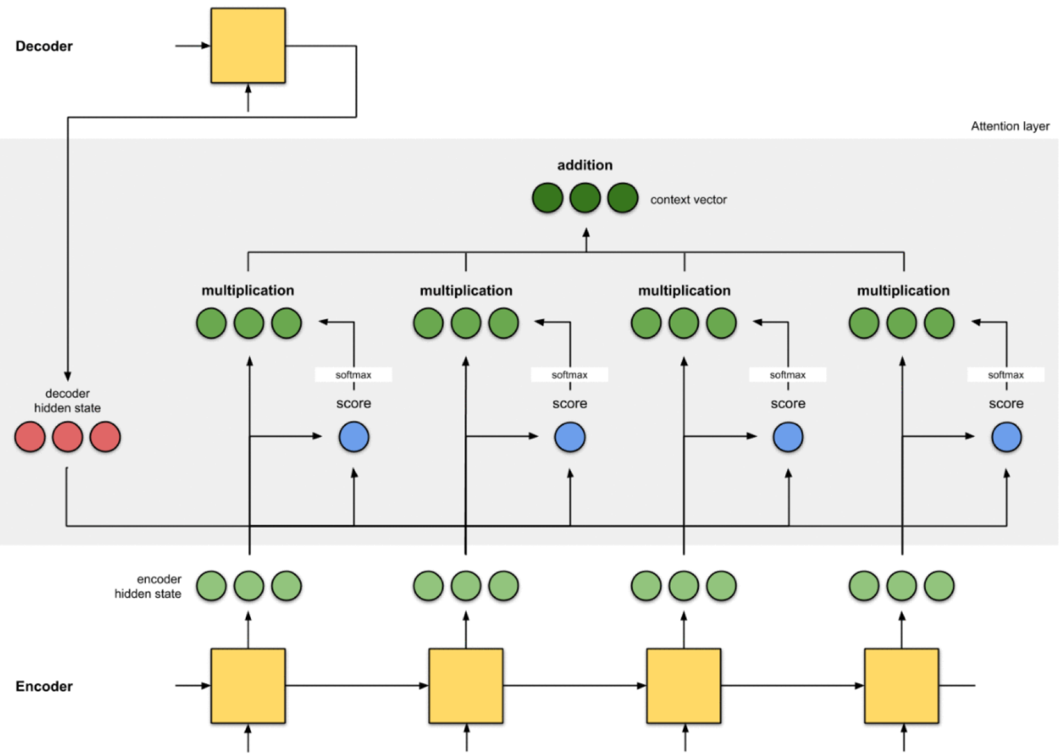
Animation from: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Seq2Seq + Attention



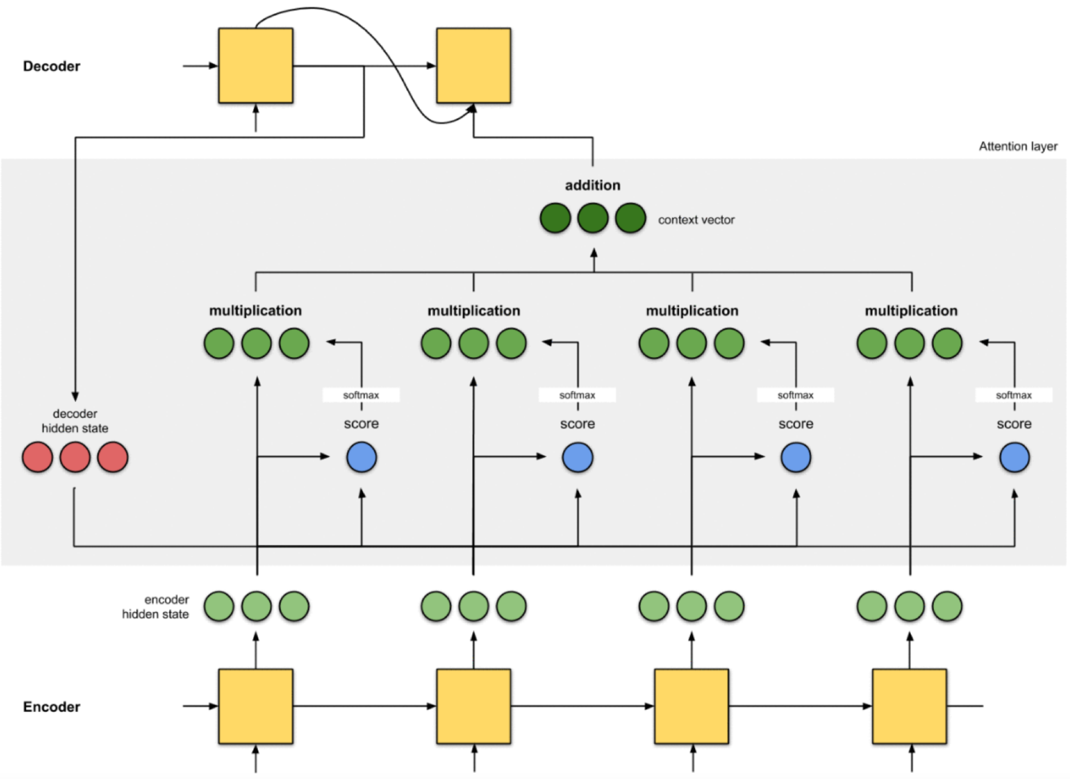
Animation from: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Seq2Seq + Attention



Animation from: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Seq2Seq + Attention



Animation from: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

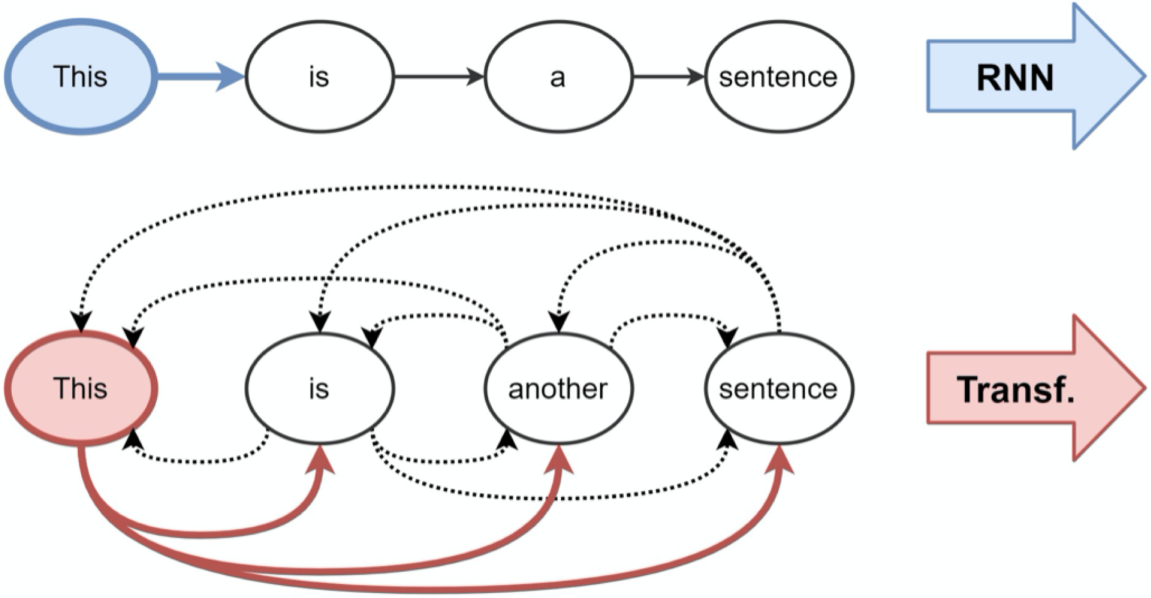
Transformers

- What if we could get rid of the recurrent architecture and use only attention?
- All the memory problems of RNNs could disappear
- No RNN, no CNN, just attention!

- Current state-of-the-art in NLP!

A. Vaswani et al. „Attention is all you need". NeurIPS 2017

Transformers



Translation?

Sentiment?

Next word?

Part-of-speech tags?

Image: https://graphdeeplearning.github.io/post/transformers-are-gnns/

Transformers

- Wait, what does this remind you of?

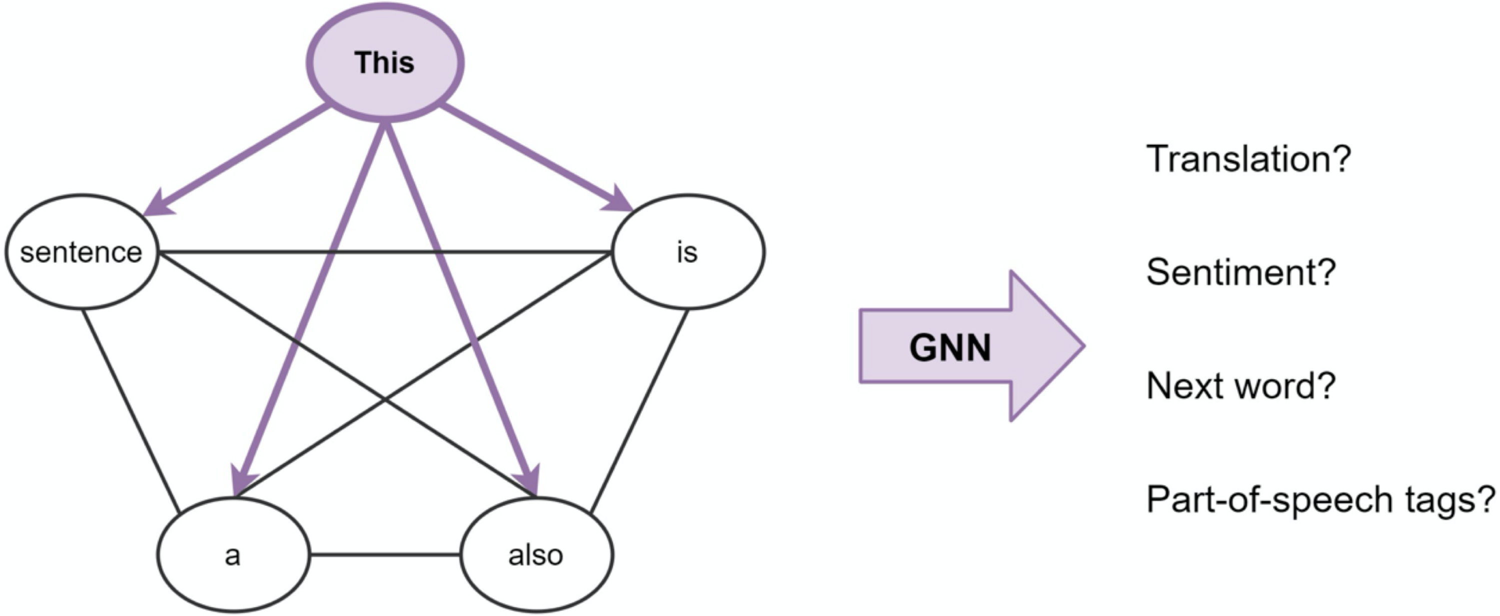


Image: https://graphdeeplearning.github.io/post/transformers-are-gnns/

Transformers

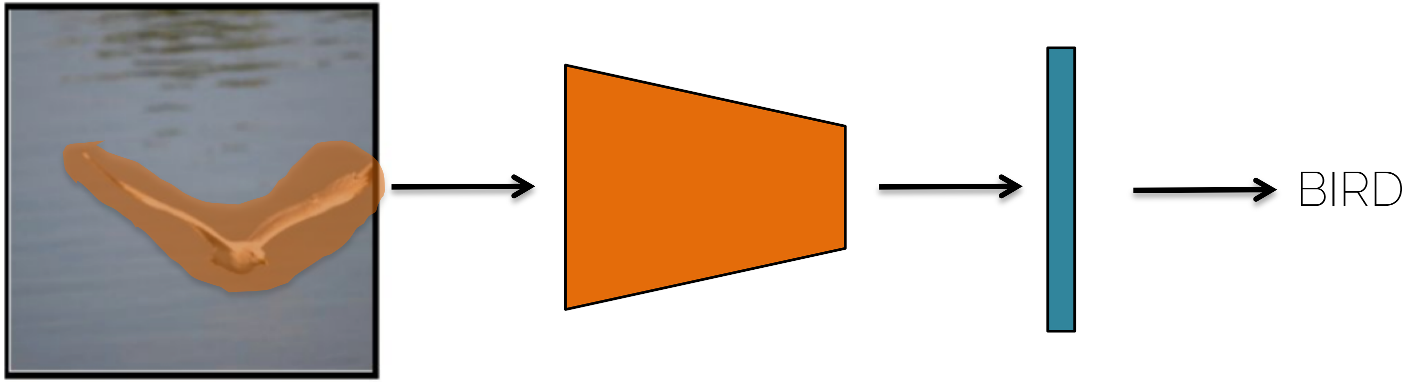
- Broadly speaking, Transformers are based on Graph Attention Networks (GAT)
- GAT replace the aggregation operation of GNN (usually a summation) by a weighted sum, i.e., an attention mechanism

Image: https://graphdeeplearning.github.io/post/transformers-are-gnns/

Attention for vision

Why do we need attention?

- We use the whole image to make the classification

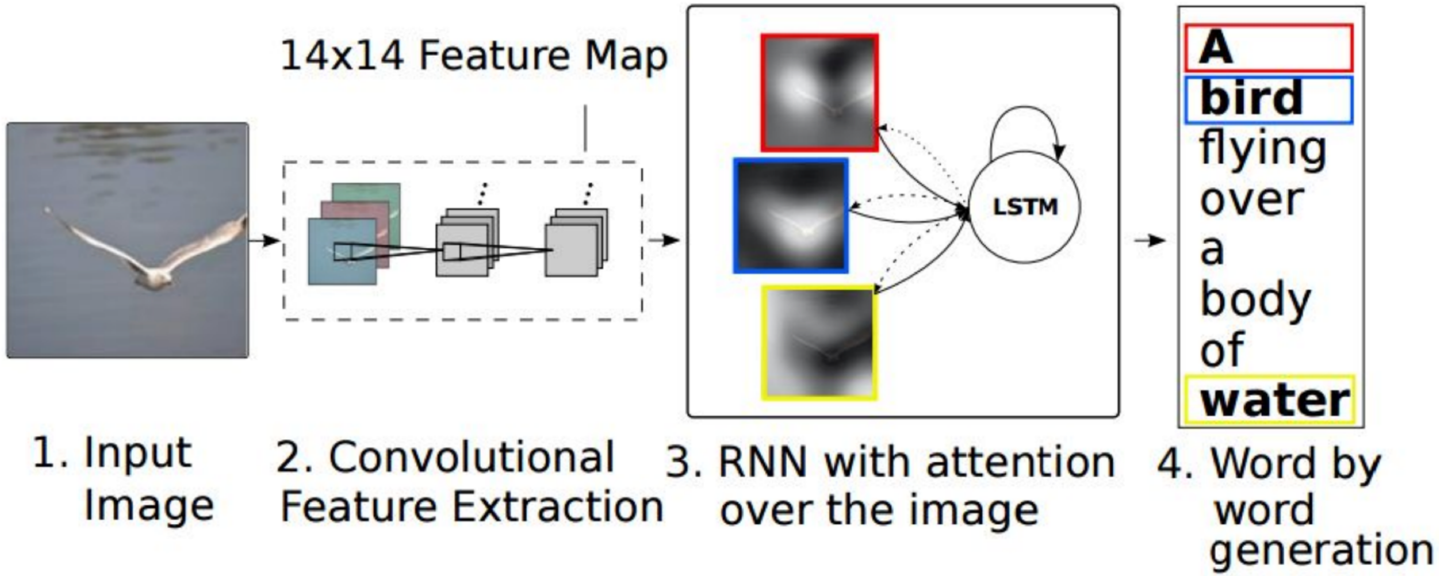


- Are all pixels equally important?

Why do we need attention?

- Wouldn't it be easier and computationally more efficient to just run our classification network on the patch?

Image captioning

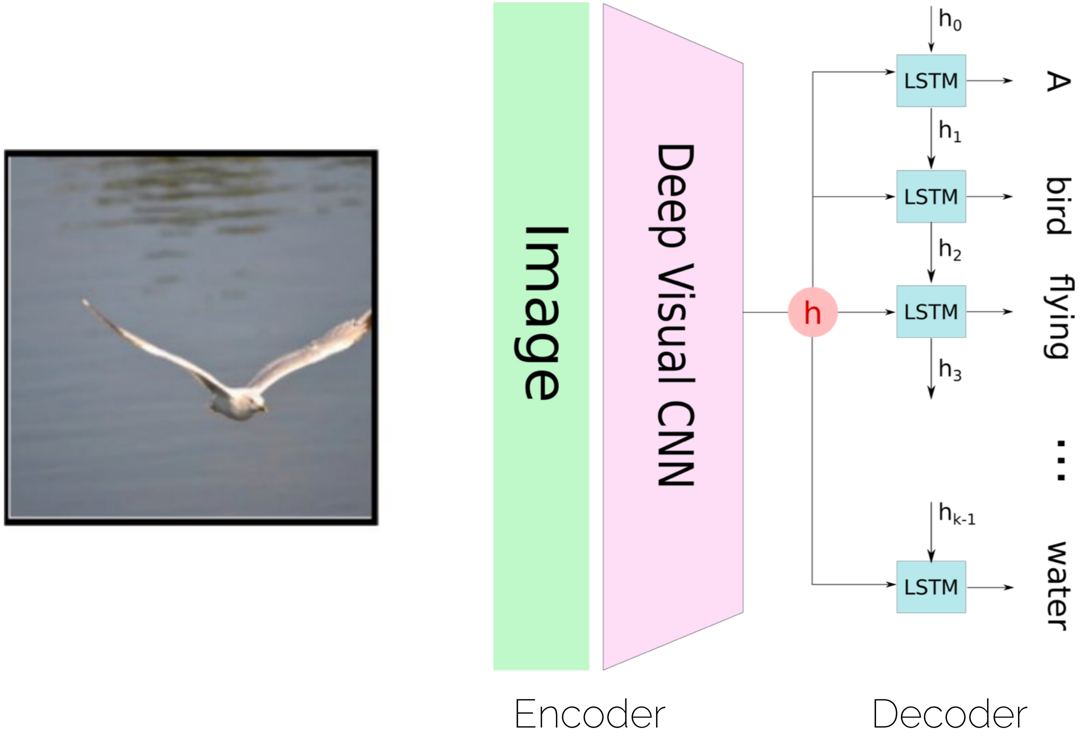


Xu et al 2015. Show attention and tell: neural image caption generation with visual attention.

Image captioning

- Input: image
- Output: a sentence describing the image.
- Encoder: a classification CNN (VGGNet, AlexNet). This computes a feature maps over the image.
- Decoder: an attention-based RNN
  - In each time step, the decoder computes an attention map over the entire image, effectively deciding which regions to focus on.
  - It receives a context vector, which is the weighted average of the conv net features.

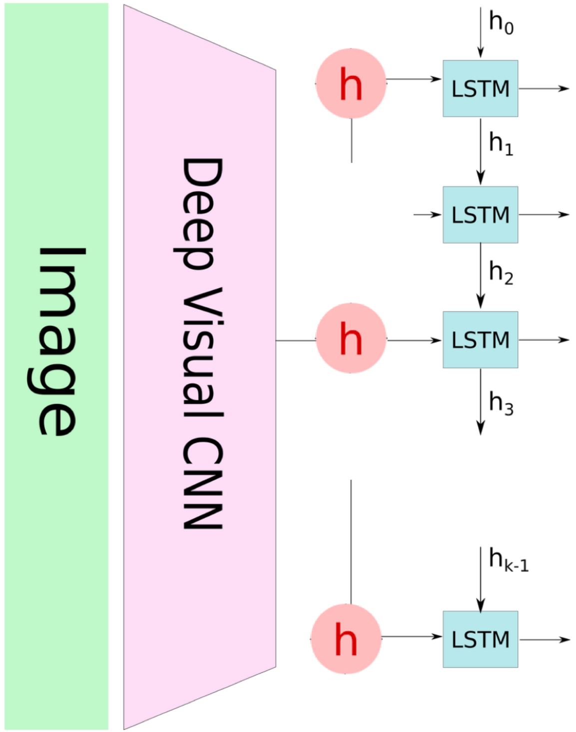
Conventional captioning



LSTM only sees the image once!

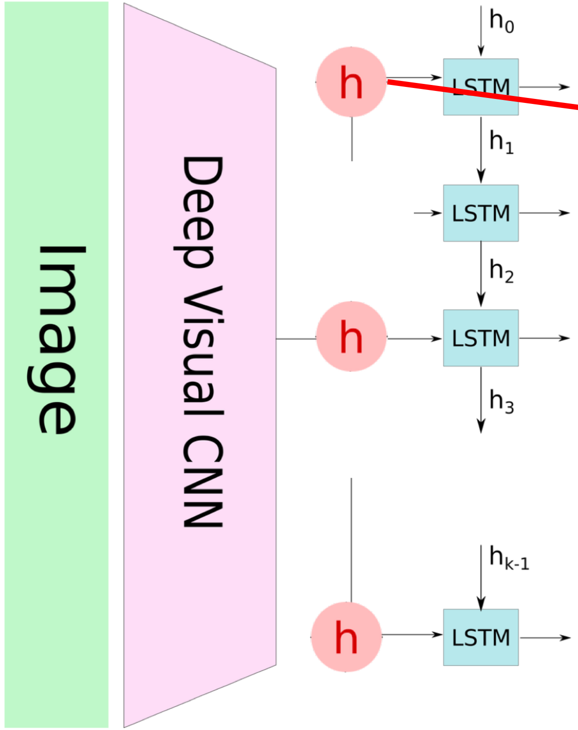
Image from: https://blog.heuritech.com/2016/01/20/attention-mechanism/

Attention mechanism



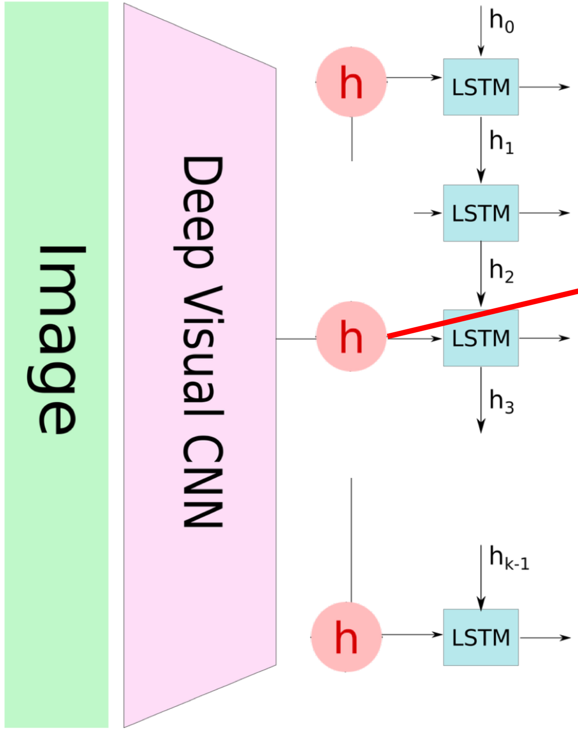
A girl is throwing a frisbee in the park

Attention mechanism



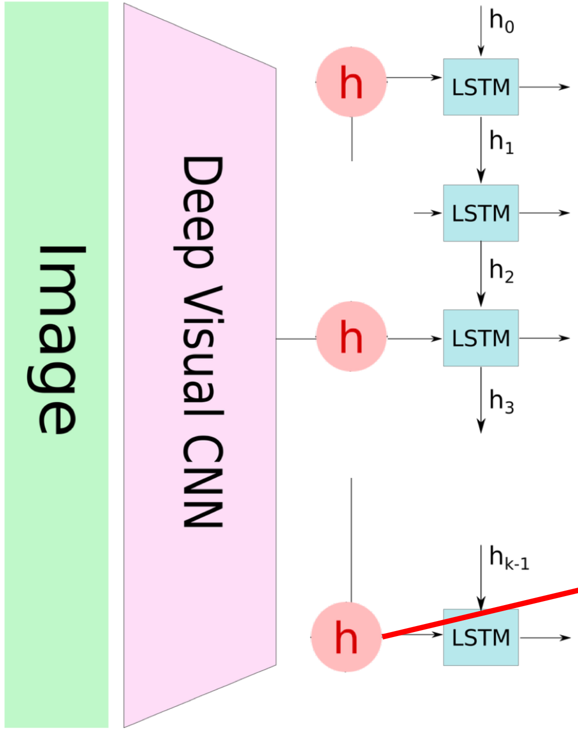
A girl is throwing a frisbee in the park

Attention mechanism



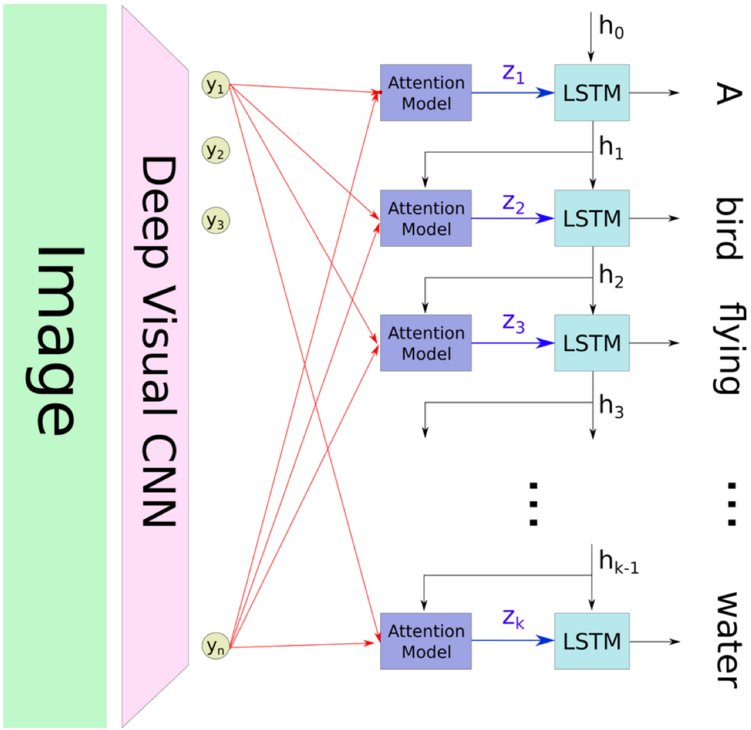
A girl is throwing a frisbee in the park

Attention mechanism



A girl is throwing a frisbee in the park

Attention mechanism

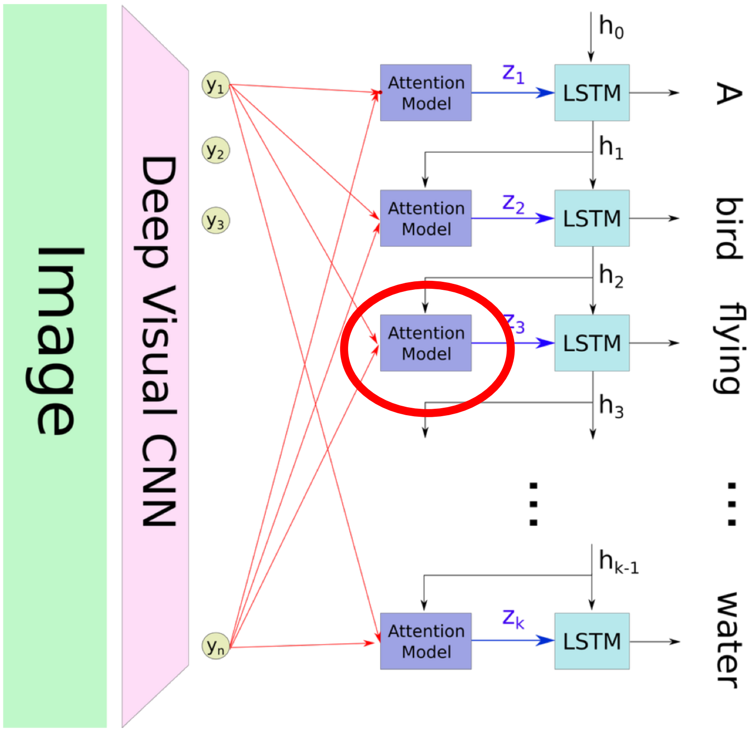


$\mathbf{y_i}$: Output of encoder are the image features which still retain spatial information (no FC layer!)

$\mathbf{Z_i}$: Output of attention model

$\mathbf{h_i}$: Hidden state of LSTM

Attention mechanism



How does the attention model look like?

Attention model

- Attention architecture

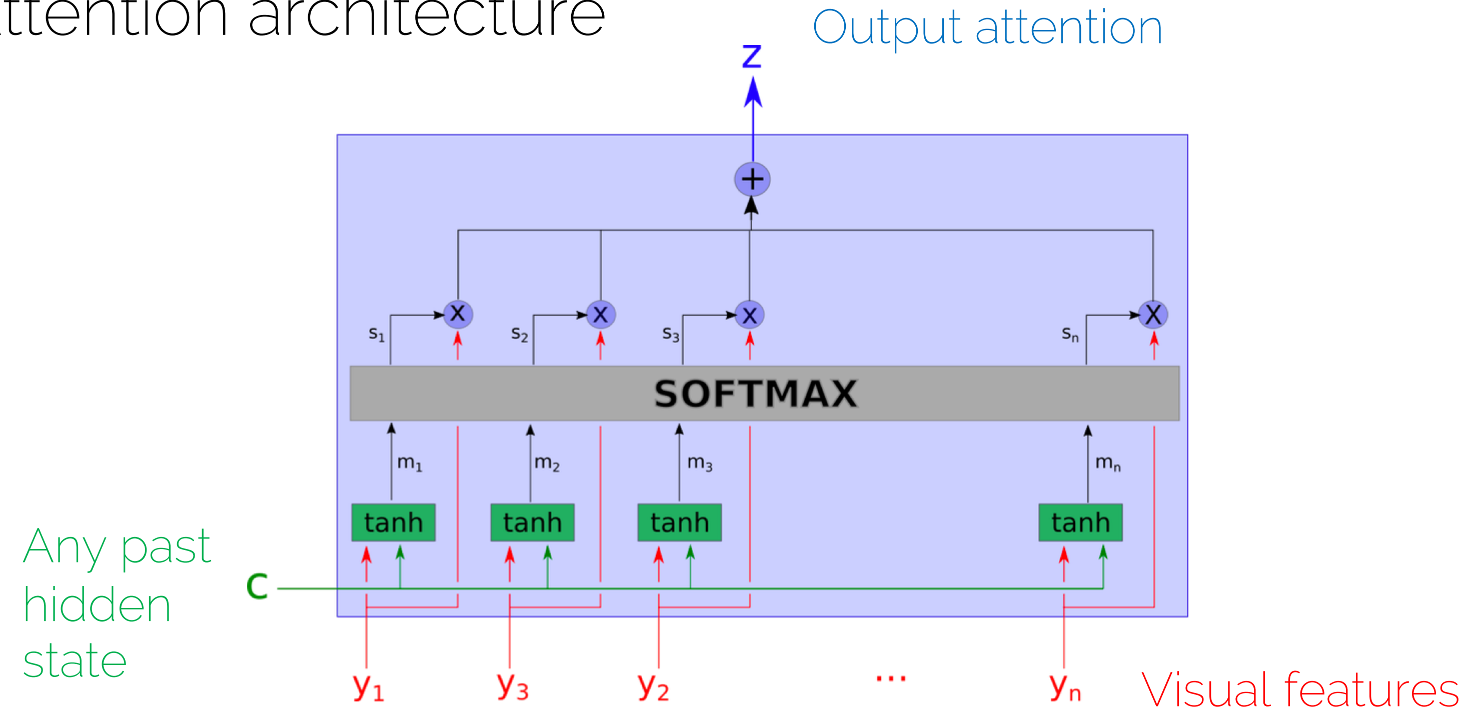
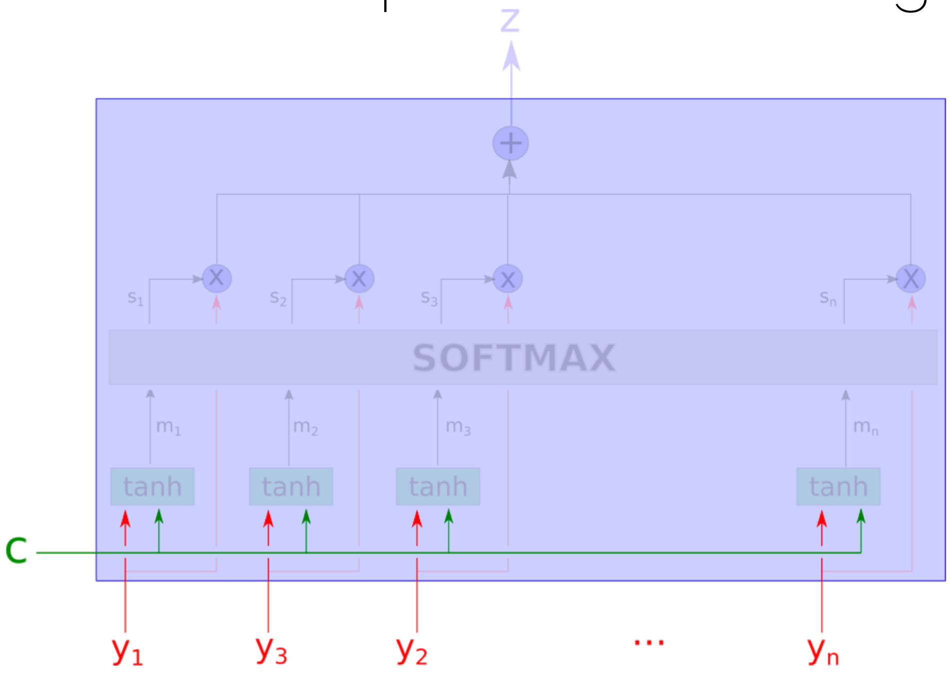


Image: https://blog.heuritech.com/2016/01/20/attention-mechanism/

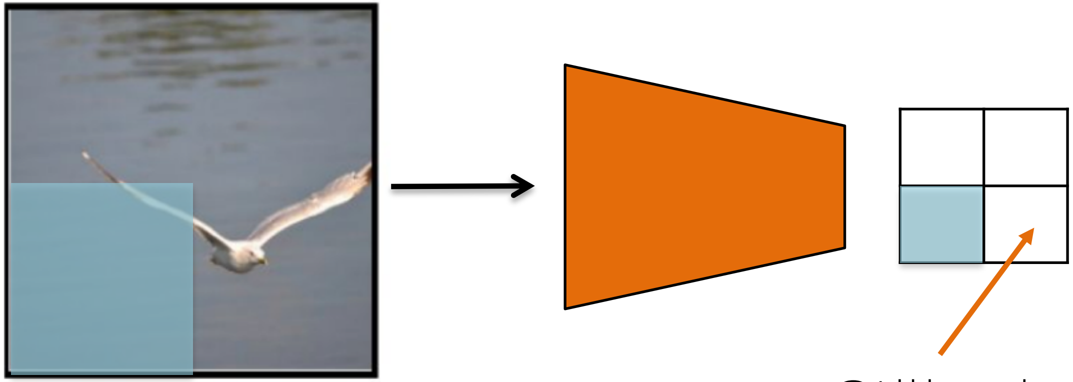
Attention model

- Inputs = feature descriptor for each image patch



Attention model

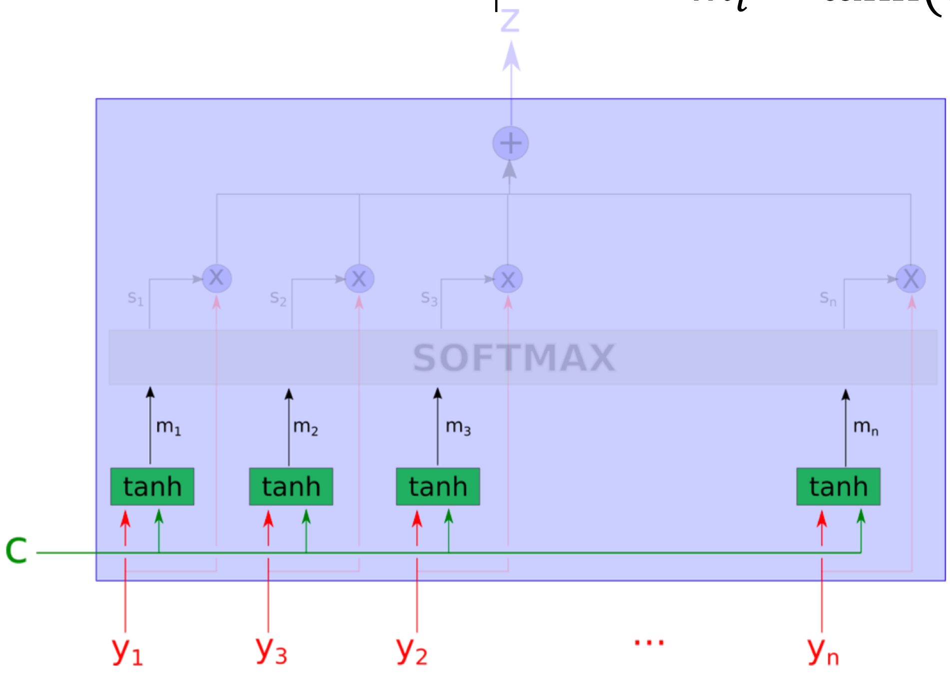
- Inputs = feature descriptor for each image patch



Still related to the spatial location of the image

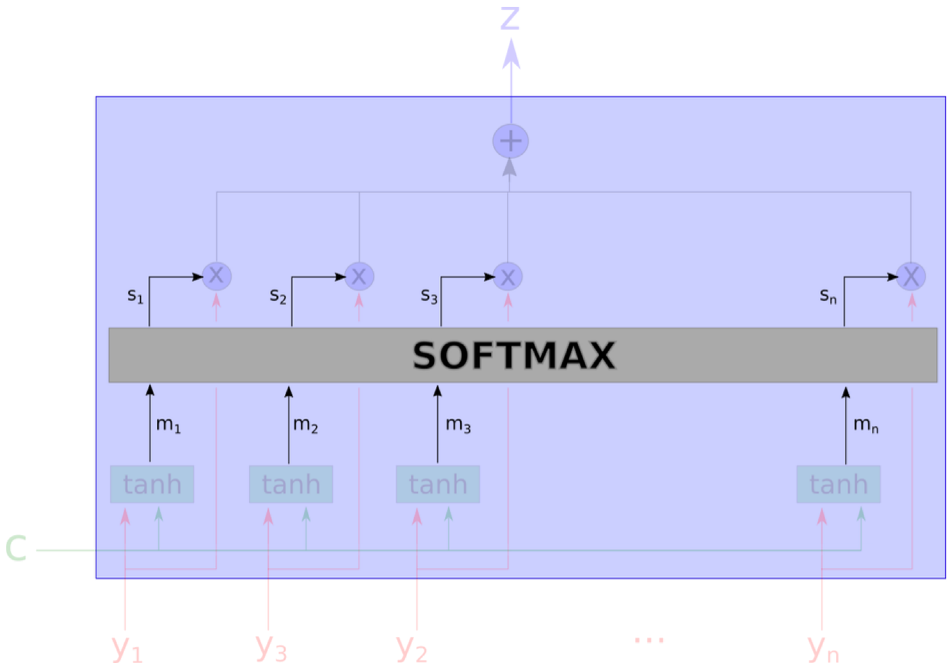
Attention model

- We want an bounded output $m_i = \tanh(W_{cm}c + W_{ym}\, y_i)$



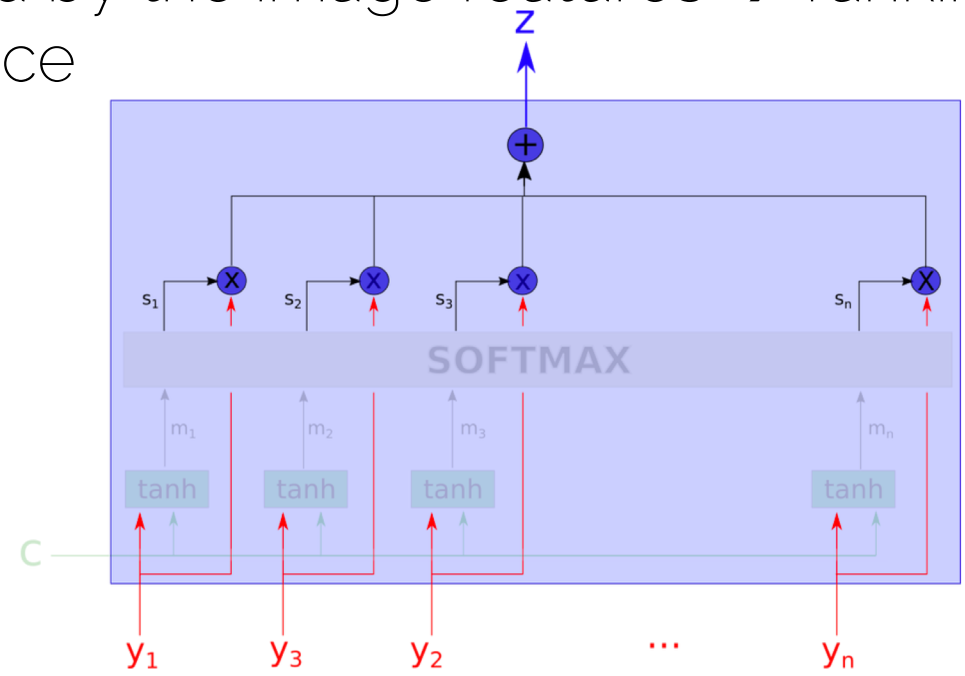
Attention model

- Softmax to create the attention values between 0 and 1



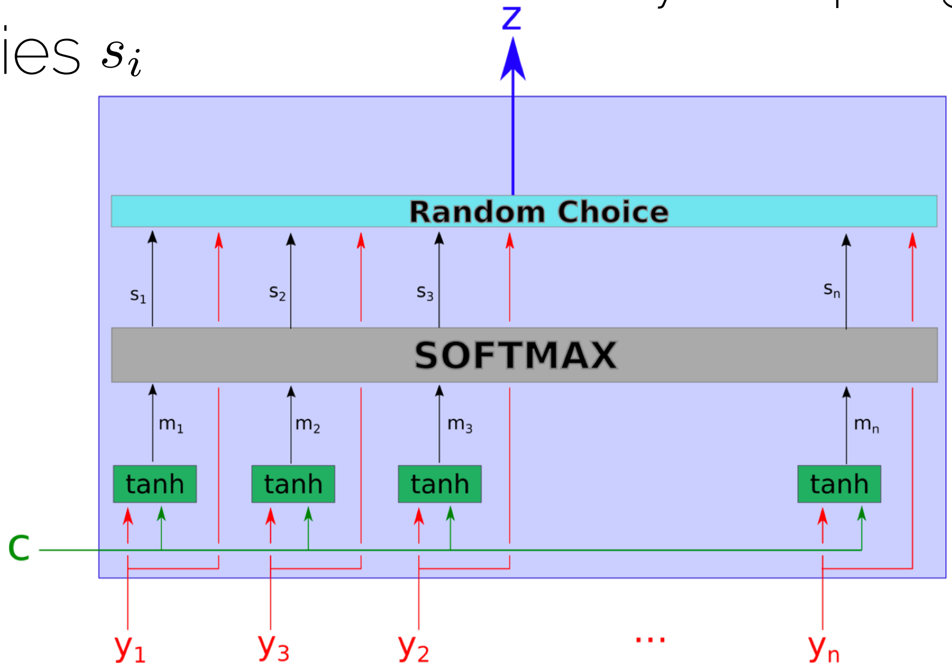
Attention model

- Multiplied by the image features → ranking by importance



Hard attention model

- Choosing one of the features by sampling with probabilities s_i



Types of attention

- **Soft attention**: deterministic process that can be backproped
- **Hard attention**: stochastic process, gradient is estimated through Monte Carlo sampling.
- Soft attention is the most commonly used since it can be incorporated into the optimization more easily

Types of attention

- Soft vs hard attention

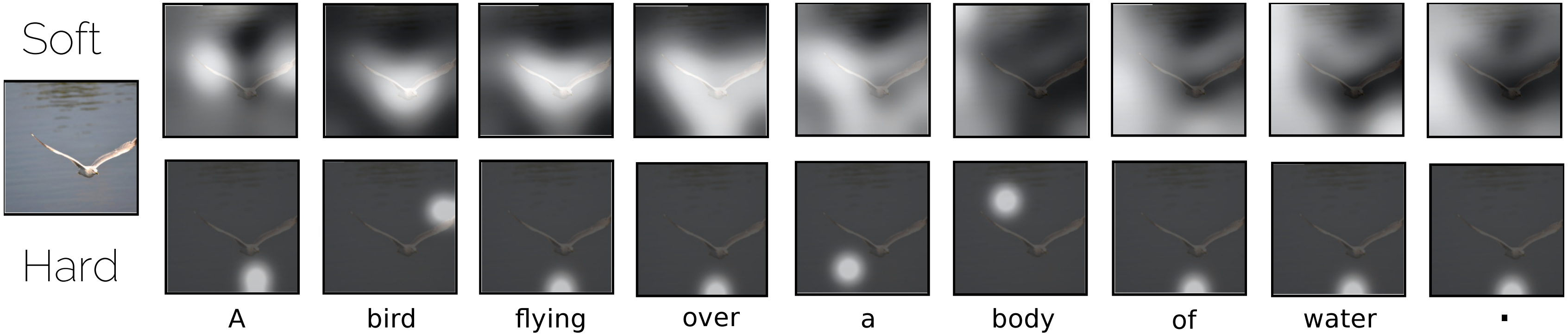
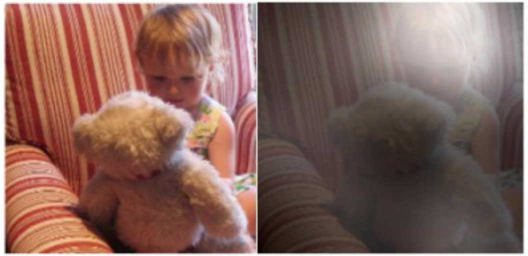


Image captioning with attention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Xu et al 2015. Show attention and tell: neural image caption generation with visual attention.

Deep Learning on graphs

Interesting works on attention

- Luong et al, "Effective Approaches to Attentionbased Neural Machine Translation," EMNLP 2015
- Chan et al, "Listen, Attend, and Spell", arXiv 2015
- Chorowski et al, "Attention-based models for Speech Recognition", NIPS 2015
- Yao et al, "Describing Videos by Exploiting Temporal Structure", ICCV 2015
- Xu and Saenko, "Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for Visual Question Answering", arXiv 2015
- Zhu et al, "Visual7W: Grounded Question Answering in Images", arXiv 2015
- Chu et al. „Online Multi-Object Tracking Using CNN-based Single Object Tracker with Spatial-Temporal Attention Mechanism". ICCV 2017