Recurrent and Graph Networks

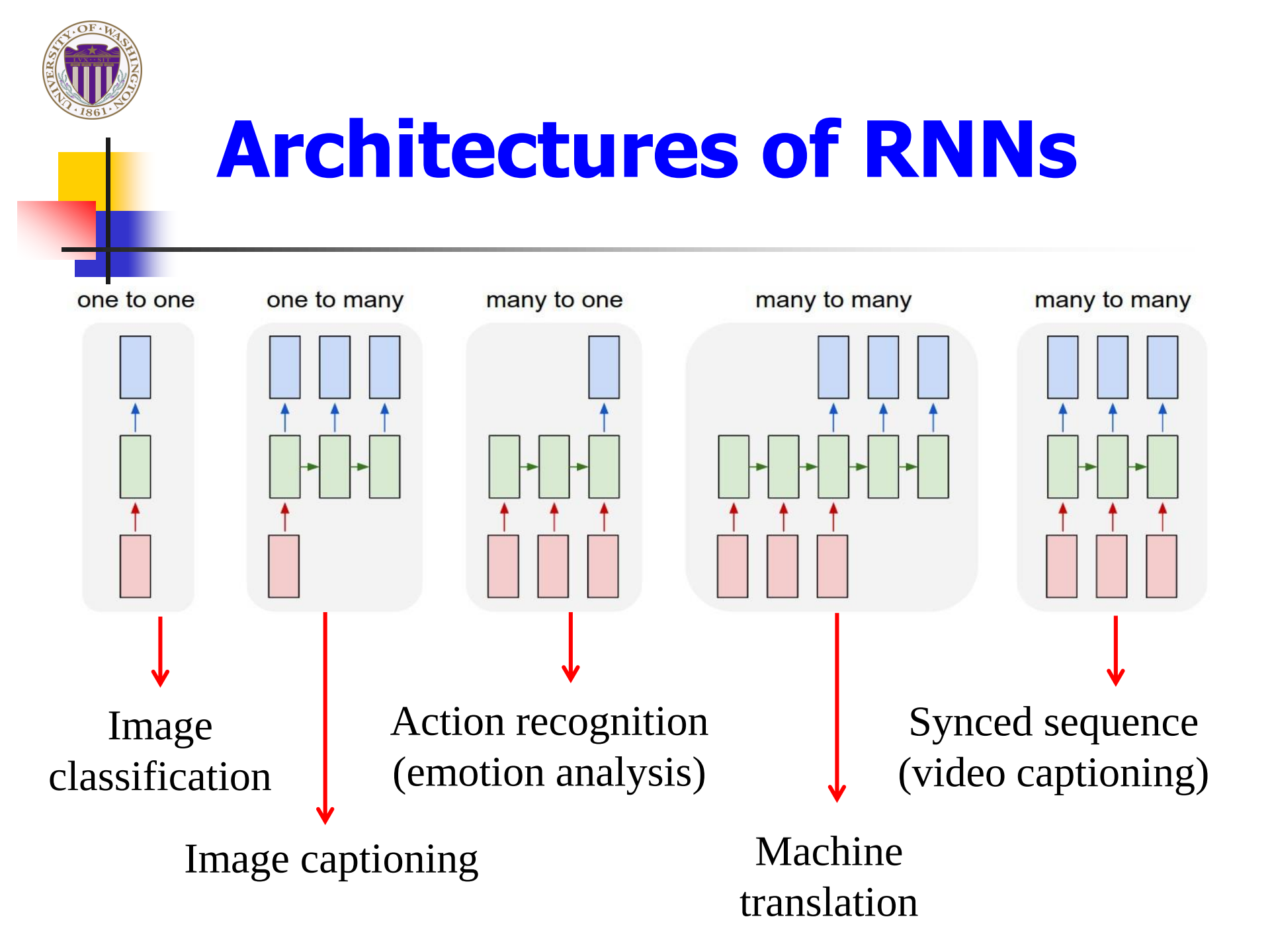
Jenq-Neng Hwang, Professor
Department of Electrical & Computer Engineering
University of Washington, Seattle WA
hwang@uw.edu

Recurrent Neural Networks (RNNs)

- RNNs (backpropagation through time, BPTT) were introduced in the late 80's.
- Hochreiter discovered the 'vanishing gradients' problem in 1991.
- Long Short Term Memory (LSTM) was published in 1997 to overcome the vanishing gradients problem
- Gated Recurrent Unit (GRU) 2015 to simplify the complexity, with comparable performance
- Hidden Markov Model (HMM) published in 1969 can be regarded as a probabilistic interpretation of a one-layer RNN

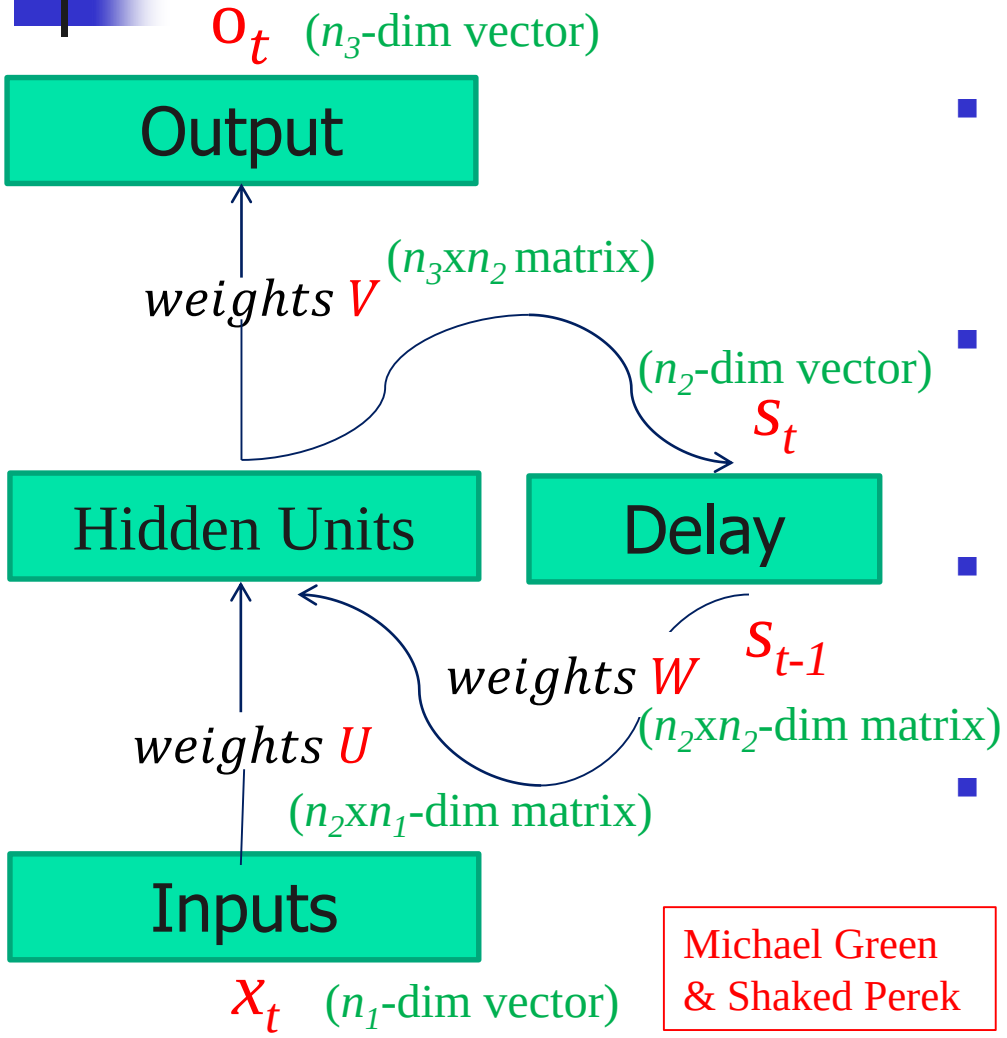
Why RNNs?

- Feed forward networks accept a fixed-sized vector (e.g., an image) as input and produce a fixed-sized vector as output
- Fixed amount of computational steps
- RNNs allow us to operate over *sequences* of vectors with varying length of inputs



Architectures of RNNs

A Simple RNN Architecture



- The hidden unit input at time t:

$$a_{ht} = Ux_t + Ws_{t-1}$$

- The hidden unit output at time t:

$$s_t = f_h(a_{ht}) = tanh(a_{ht})$$

- The output unit input at time t:

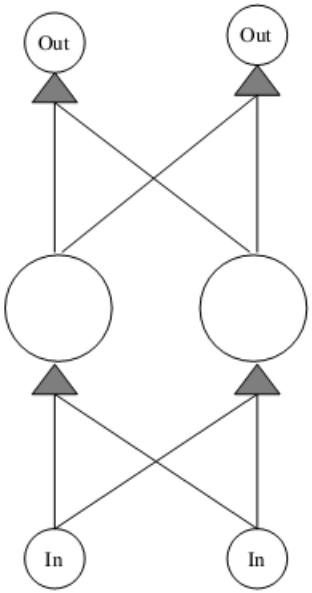
$$a_{ot} = Vs_t$$

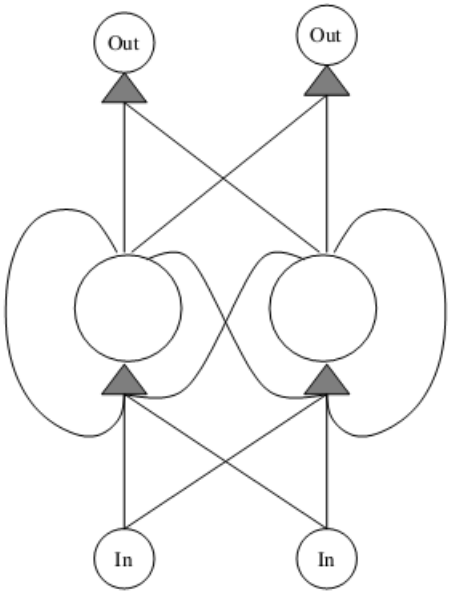
- The network output at time t is:

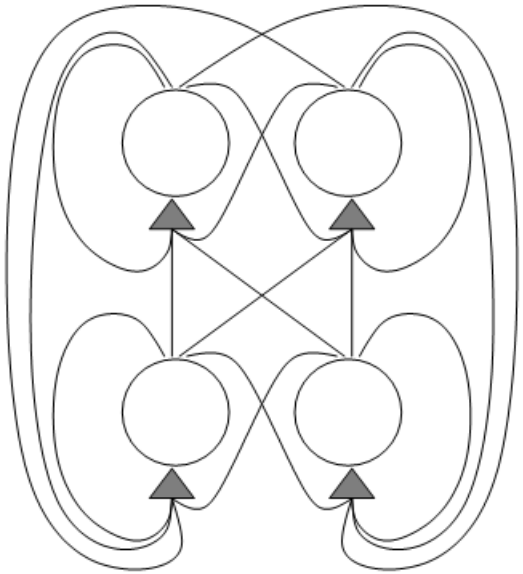
$$o_t = f_o(a_{ot}) = softmax(a_{ot})$$

Michael Green
& Shaked Perek

Recurrent Connections





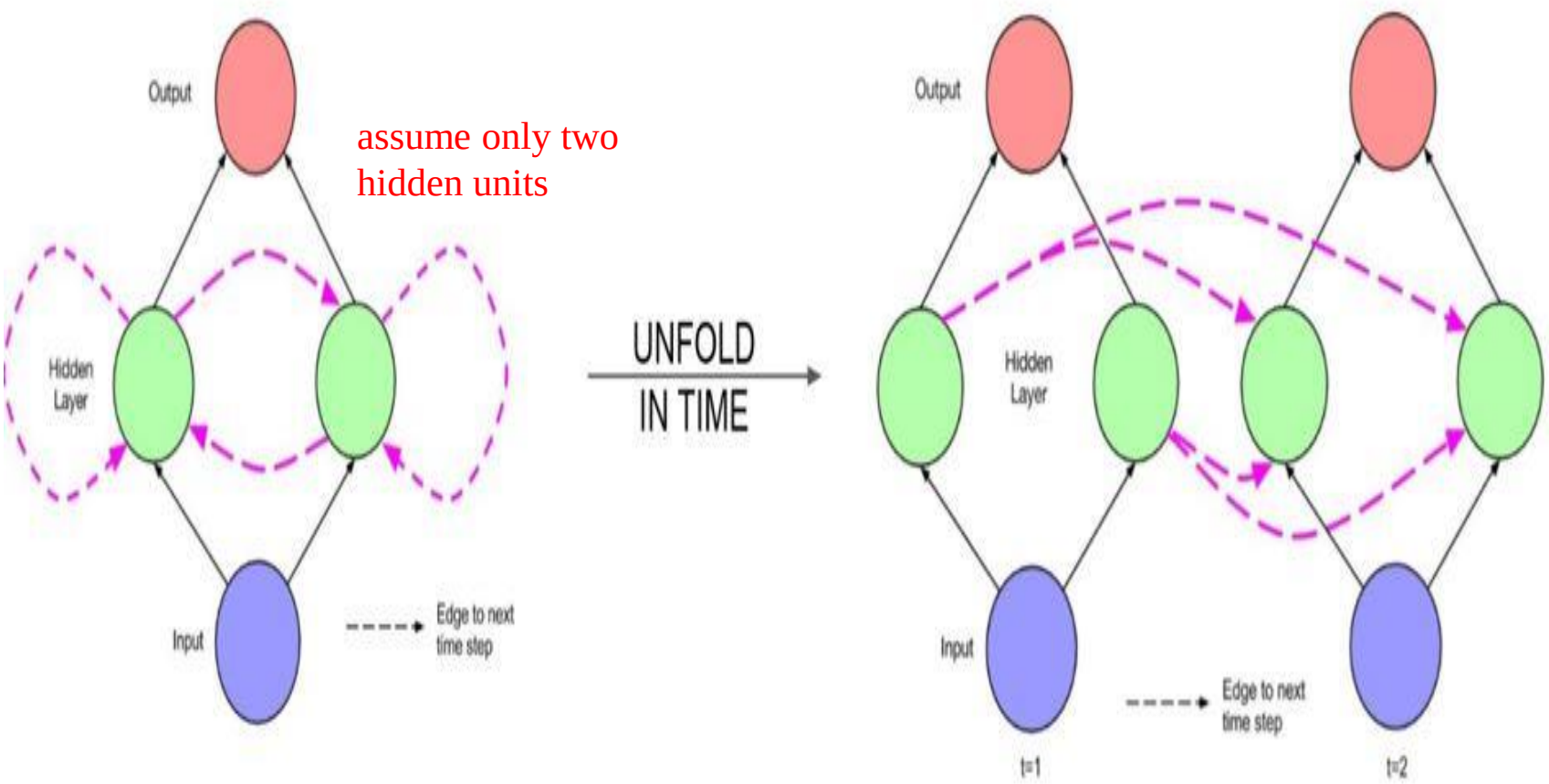


Left: feedforward neural network

Middle: a simple (hidden-layer) recurrent neural network

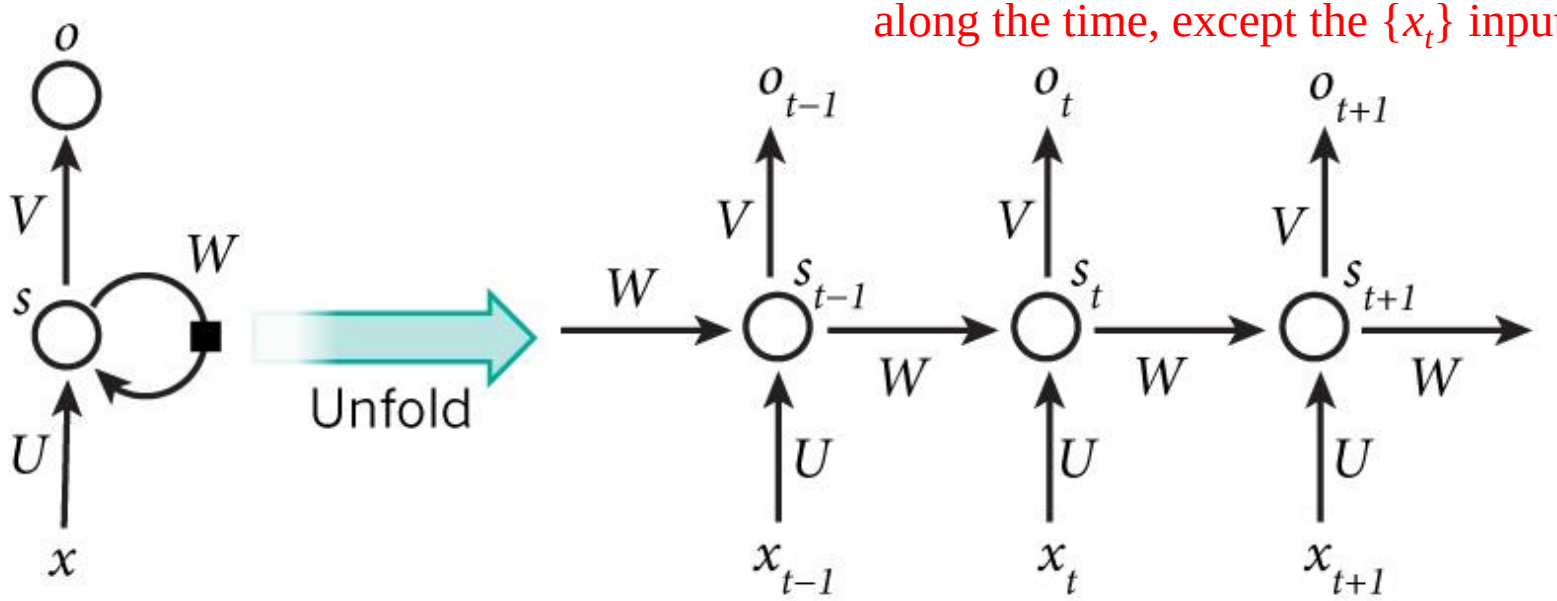
Right: fully connected recurrent neural network

Unfolding over Time

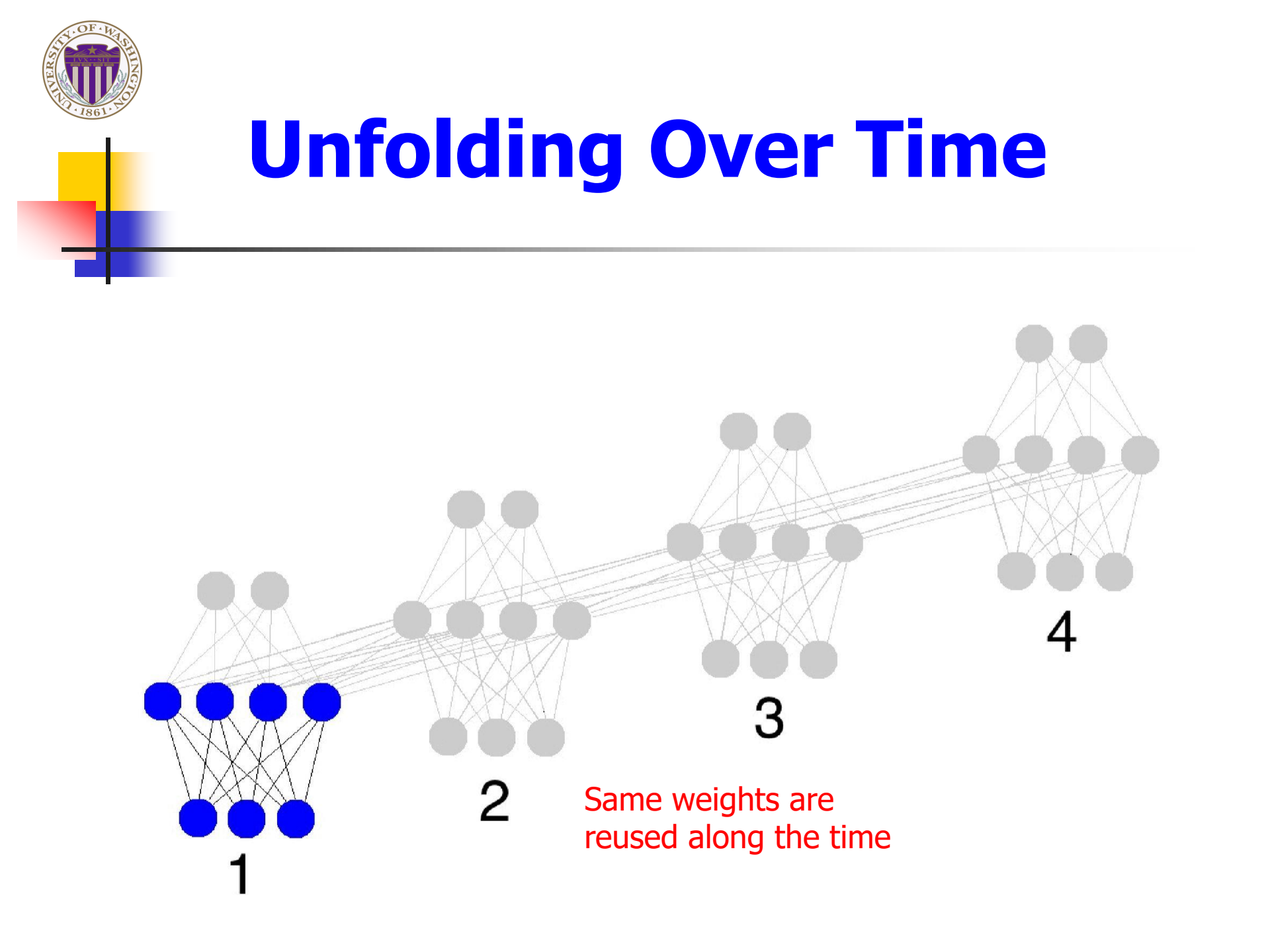


Unfolding Over Time

Same weights, U, V, W are reused along the time, except the $\{x_t\}$ inputs



- The recurrent network can be converted into a feed forward network by unfolding over time
- Making time dimension into physical layers



Unfolding Over Time

Loss Function for Training

- If a network training sequence starts at time t_0 and ends at t_1, the total **loss function** is the sum over time of the squared error (or cross entropy) function between ground truth y_t and prediction $\hat{y}_t = o_t \ , t=t_0, t_o+1, \ldots\ldots t_1$

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

$$E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t)$$

$$= -\sum_t y_t \log \hat{y}_t$$

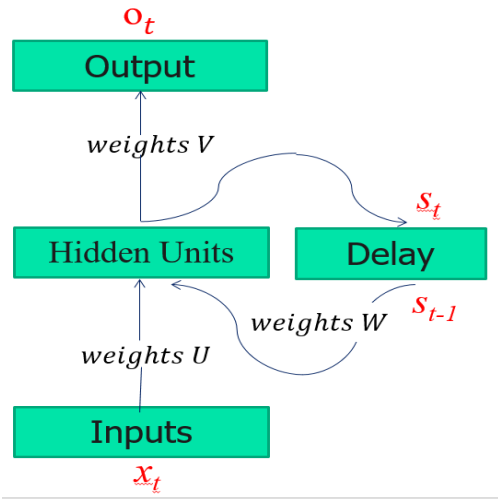
Back Propagation Through Time (BPTT)

- BPTT learning algorithm is an extension of standard backpropagation that performs gradients descent on an unfolded network.
- The gradient descent weight updates have contributions from each time step.
- The errors have to be back-propagated through time as well as through the network layers

$$V(t+1) = V(t) - \alpha \frac{\partial E}{\partial V} = V(t) - \alpha \sum_t \frac{\partial E_t}{\partial V}$$

$$U(t+1) = U(t) - \alpha \frac{\partial E}{\partial U} == U(t) - \alpha \sum_t \frac{\partial E_t}{\partial U}$$

$$W(t+1) = W(t) - \alpha \frac{\partial E}{\partial W} = W(t) - \alpha \sum_t \frac{\partial E_t}{\partial W}$$



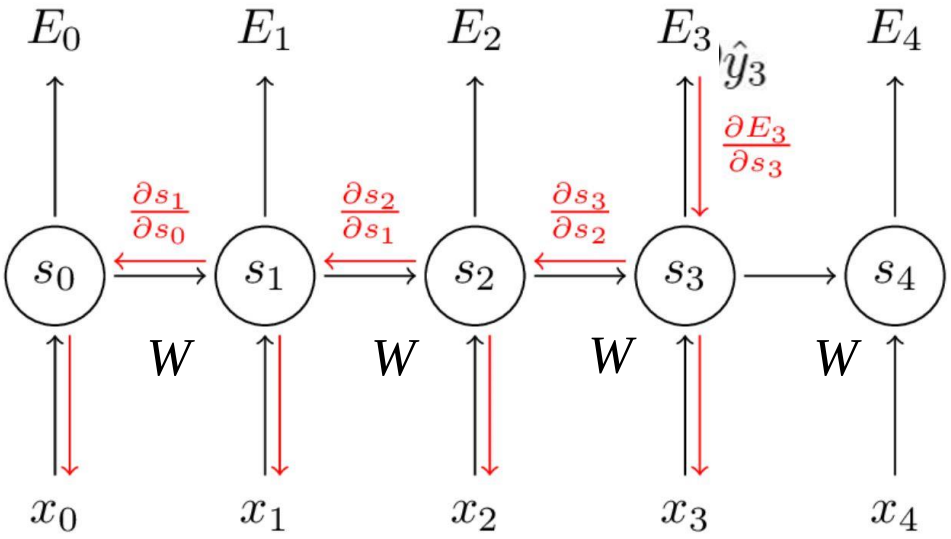
Back Propagation Through Time (BPTT)

$$\hat{y}_t = o_t \ , \ t=t_0, t_o+1, \ldots . t_1$$

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

$$E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t)$$

$$= -\sum_t y_t \log \hat{y}_t$$

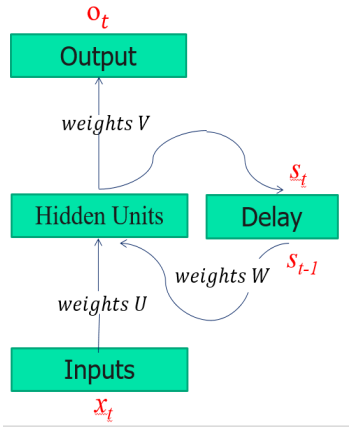


$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W}$$

$$s_3 = \tanh(U x_3 + W s_2)$$

$$\frac{\partial E_3}{\partial W} = \sum_{K=1}^{3} \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial s_k}\frac{\partial s_k}{\partial W}$$

Paul Werbos, "Generalization of backpropagation with application to a recurrent gas market model,". *Neural Networks, 1988*



$$\frac{\partial E_3}{\partial W} = \sum_{K=1}^{3} \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\left(\prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}}\right)\frac{\partial s_k}{\partial W}$$

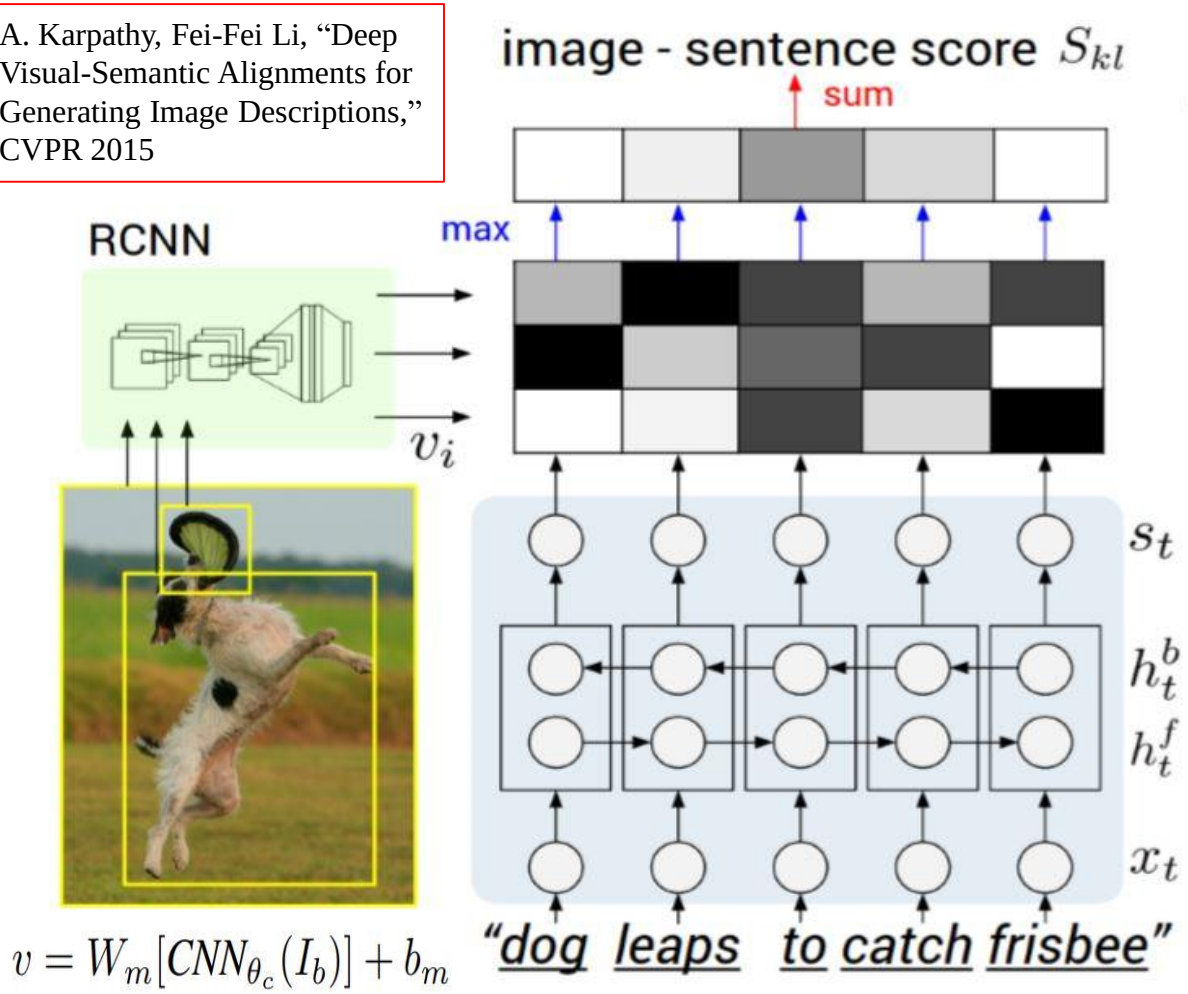
$$\frac{\partial s_3}{\partial s_1} = \frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial s_1}$$

RNN for Image Description

- Datasets of image captions are available (e.g., Microsoft CoCo)
- These descriptions mention of several entities whose locations in the images are unknown
- A multimodal RNN to infer the latent alignment between segments of sentences and the region of the image (identified by R-CNN) that they describe.
- The objective is to take an input image and generate its description in text.

Image to Text Aligment

A. Karpathy, Fei-Fei Li, "Deep Visual-Semantic Alignments for Generating Image Descriptions," CVPR 2015



$$S_{kl} = \sum_{t \in g_l} \sum_{i \in g_k} max(0, v_i^T s_t)$$

- g_k is the set of image fragments in image k
- g_l is the set of sentence fragments in sentence l.
- Each s_t can find best v_i

Bi-Directional RNN

(h_t hiddent units, s_t output units, same dim as v_i)

300-dim word2vec

Image/Text Data Embedding

- R-CNN(I_b) transforms the image pixels inside bounding box I_b into 4096-dimensional activations of the fully connected layer immediately before the classifier.
- The matrix Wm has dimensions $h \times 4096$, where h is the size of the multimodal embedding space (like PCA projection, were h ranges from 1000-1600 in the experiments). Every detected object is thus represented as a set of h-dimensional vectors $\{v_i \mid i = 1 \ldots 20\}$, 19 detected locations + image.
- The h-dimensional representation s_t for the t-th word is a function of both the word at that location and also its surrounding context in the sentence.

Training & Testing Datasets

- Flickr8K, Flickr30K and MSCOCO datasets, with 8,000, 31,000 and 123,000 images respectively.
- Each image is annotated with 5 sentences using Amazon Mechanical Turk (MTurk, crowdsourcing) -- 2538, 7414, and 8791 words for Flickr8k, Flickr30K, and MSCOCO datasets respectively.
- For Flickr8K and Flickr30K, 1,000 images for validation, 1,000 for testing and the rest for training.
- For MSCOCO, 5,000 images for both validation and testing.

Sentence Retrieving

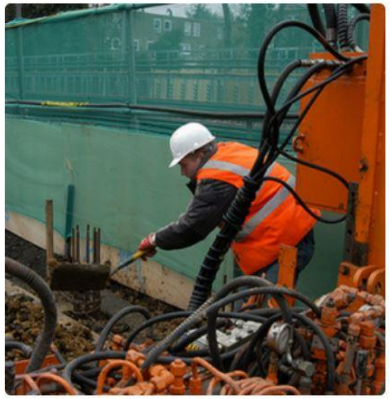
- Given a set of images and sentences, retrieve sentences from an image query by sorting based on the image-sentence score S_{kl}
- The model generates sensible descriptions of images by retrieving sentences from training sets based on associated scores ($v^T_i s_t$).
- Comparing most similar training set image as determined by L2 norm over VGGNet fc7 features.

	Flickr8K				Flickr30K				MSCOCO 2014					
Model	B-1	B-2	B-3	B-4	B-1	B-2	B-3	B-4	B-1	B-2	B-3	B-4	METEOR	CIDEr
Nearest Neighbor	—	—	—	—	—	—	—	—	48.0	28.1	16.6	10.0	15.7	38.3
Mao et al. [38]	58	28	23	—	55	24	20	—	—	—	—	—	—	—
Google NIC [54]	63	41	27	—	66.3	42.3	27.7	18.3	66.6	46.1	32.9	24.6	—	—
LRCN [8]	—	—	—	—	58.8	39.1	25.1	16.5	62.8	44.2	30.4	—	—	—
MS Research [12]	—	—	—	—	—	—	—	—	—	—	—	21.1	20.7	—
Chen and Zitnick [5]	—	—	—	14.1	—	—	—	12.6	—	—	—	19.0	20.4	—
Our model	57.9	38.3	24.5	16.0	57.3	36.9	24.0	15.7	62.5	45.0	32.1	23.0	19.5	66.0

Table 2. Evaluation of full image predictions on 1,000 test images. **B-n** is BLEU score that uses up to n-grams. High is good in all columns.

Sentence Retrieving

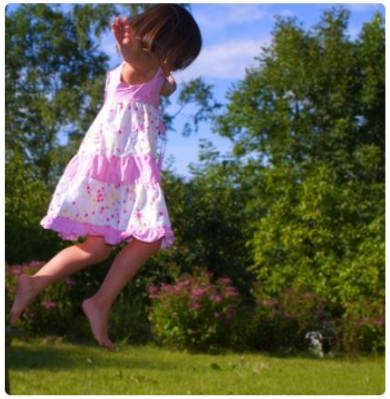
"man in black shirt is playing guitar."



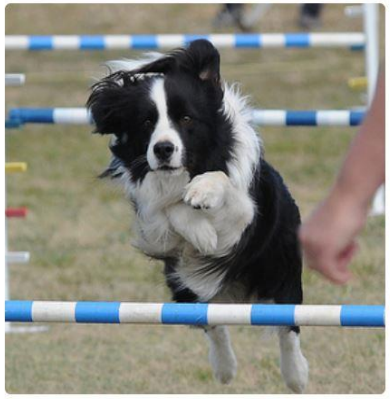
"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

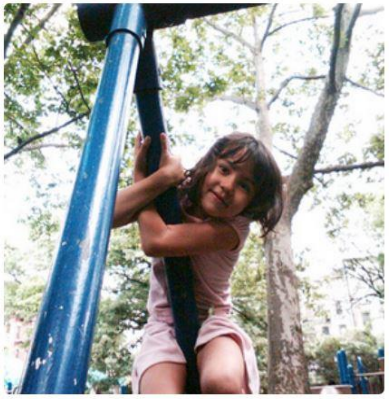
"boy is doing backflip on wakeboard."



"girl in pink dress is jumping in air."



"black and white dog jumps over bar."

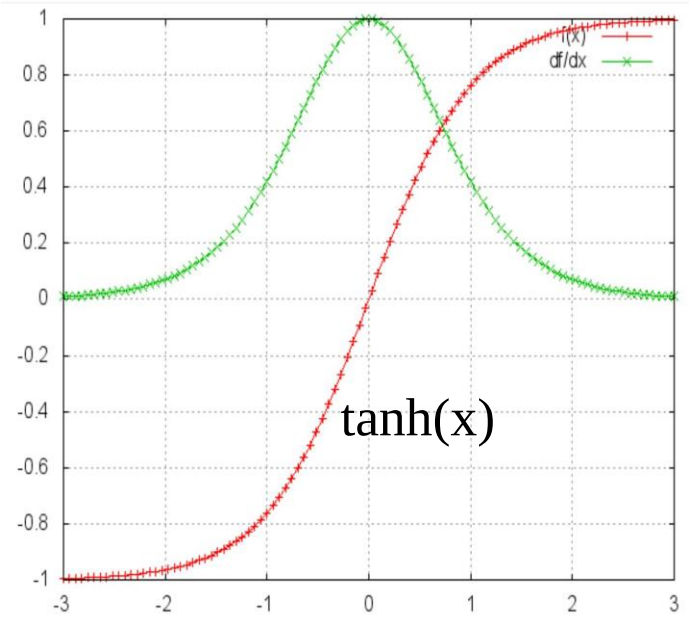


"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

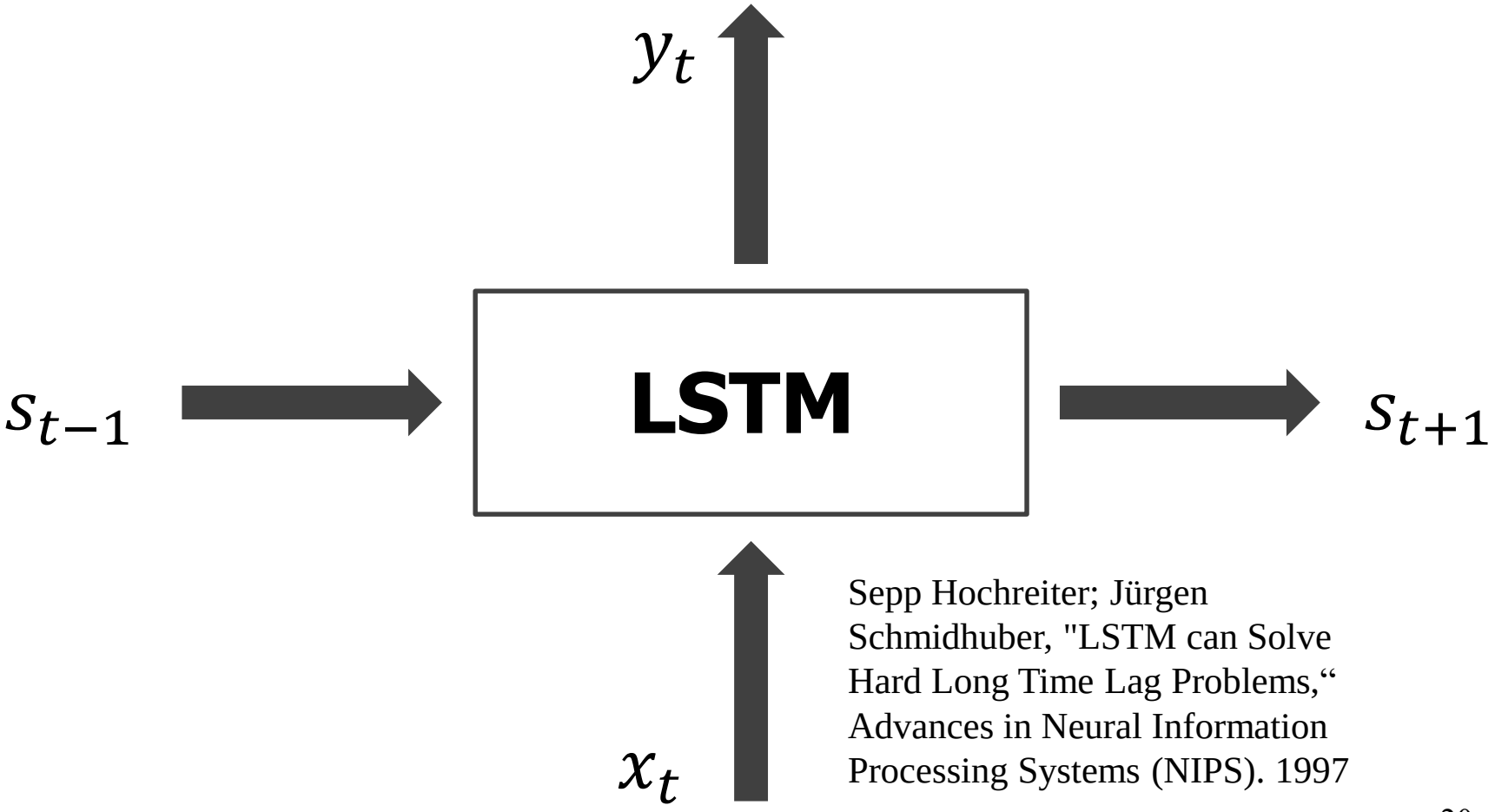
Long Short-Term Memory (LSTM)

- LSTM was invented to solve the vanishing gradients problem in an RNN – too many sigmoid derivatives multiplied together (little update in early layers)
- LSTM maintain a more constant error flow in the backpropogation process via gating.
- LSTM can learn over more than 1000 time steps (long-term memory) , and thus can handle long sequences that are linked remotely.



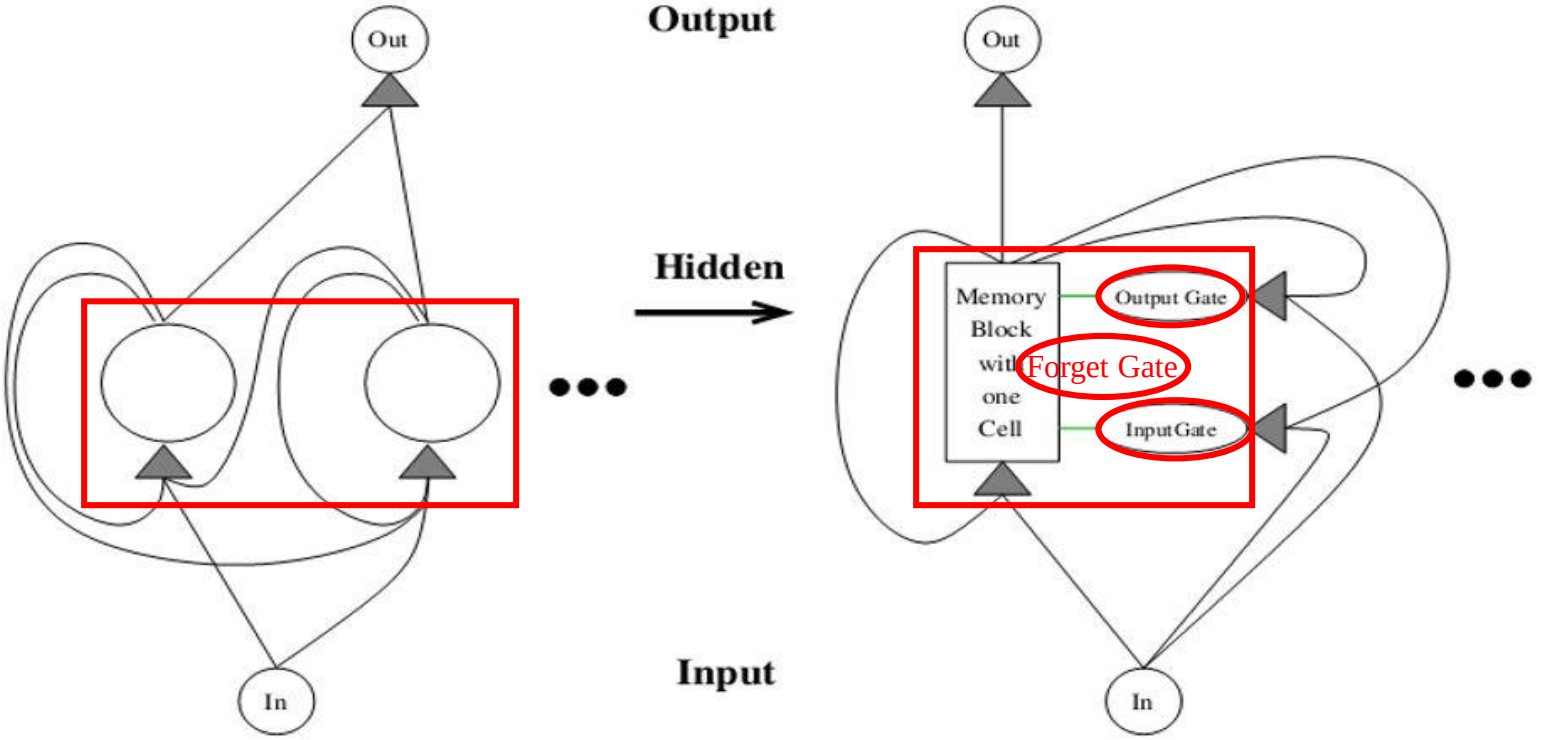
$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \left(\prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}$$

LSTM – Similar to RNN



Sepp Hochreiter; Jürgen Schmidhuber, "LSTM can Solve Hard Long Time Lag Problems," Advances in Neural Information Processing Systems (NIPS). 1997

An LSTM Architecture

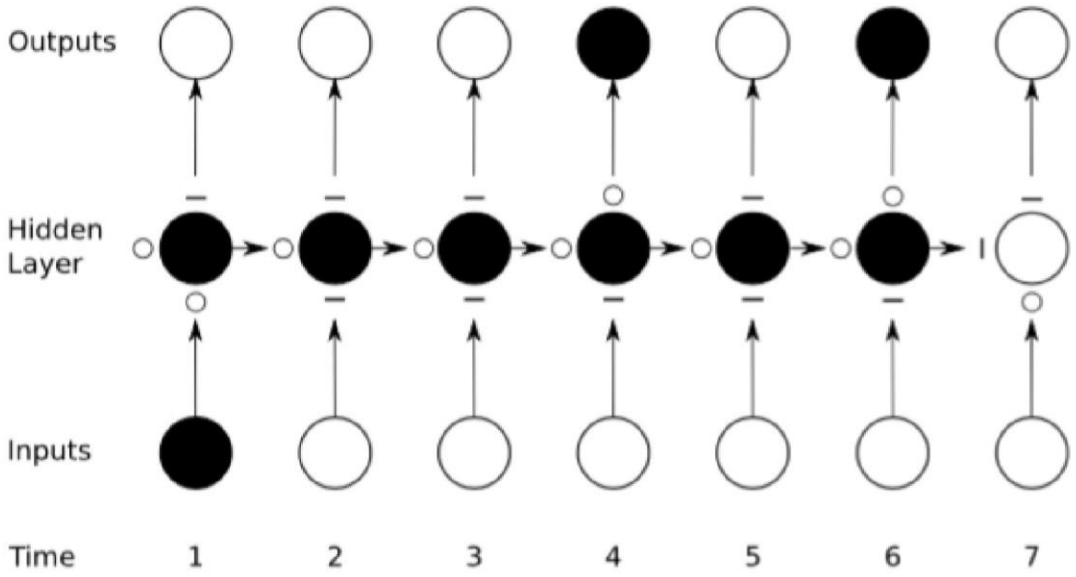


An RNN with one fully recurrent hidden layer (2 hidden units)

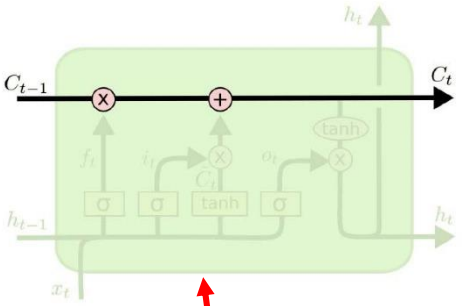
An LSTM network (one cell can have multiple hidden neurons) with memory blocks in the hidden layer

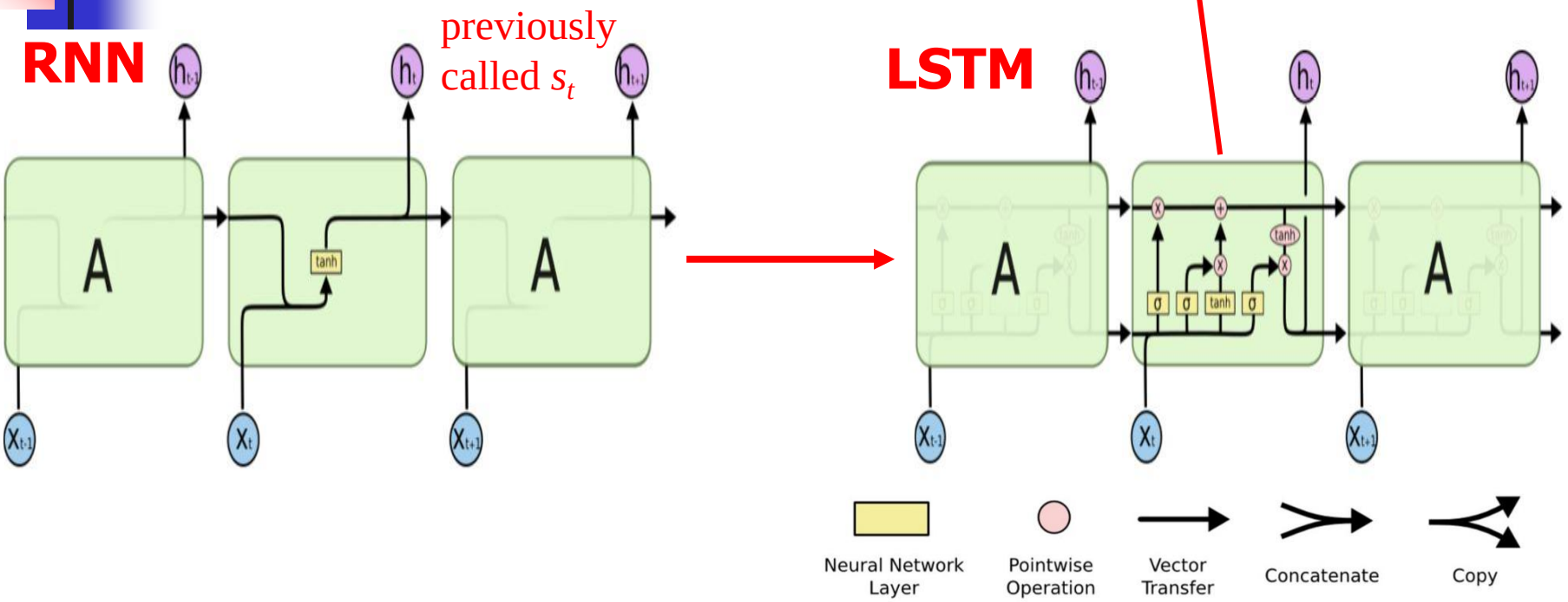
Why Gated Input/Output

- Dark nodes indicate their sensitivities to first input
- Three gates (input, forget, output) are either open/closed
- The dark memory cells "remember" the first input as long as forget gate is open (forward) and input gates are closed



An LSTM Architecture



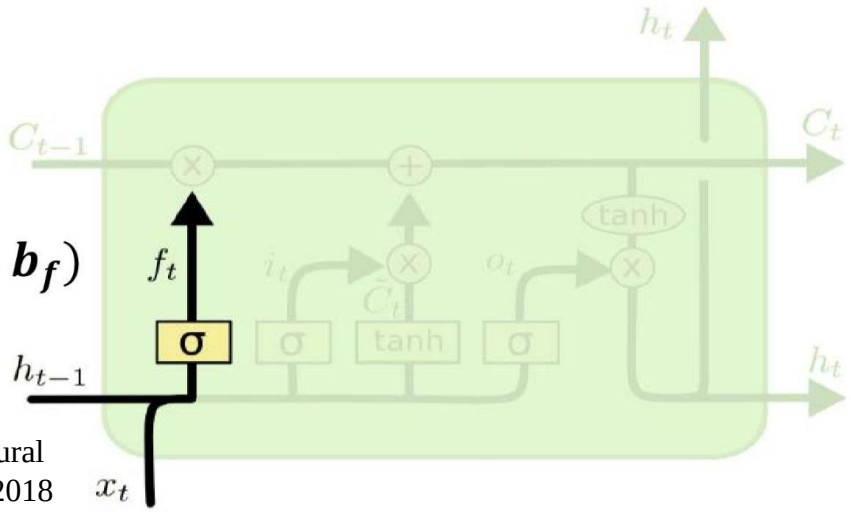


- Maintains a (cell memory) vector C_t, the same dimensionality as the hidden state, h_t (previously called s_t)
- Memory can be deleted or added from this cell vector via the forget & input gates (controlled by h_{t-1} and x_t).

Forget Gate

- Decides what information to throw away or remember from the previous cell state (scalar or vector)
- Based on previous hidden state h_{t-1} and input x_t, and outputs a number between 0 and 1 (sigmoid), how much to retain the value of cell state C_{t-1} (0 being forget, 1 being keep).

$$\boldsymbol{f_t = sigmoid(\theta_{xf} x_t + \theta_{hf} h_{t-1} + b_f)}$$



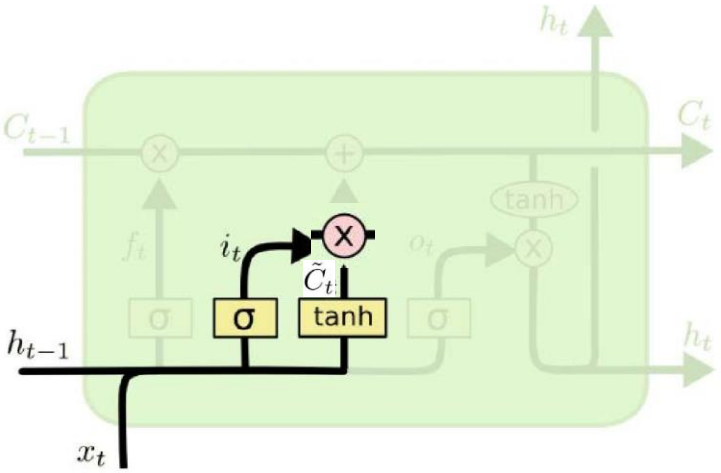
Adapted from Pankaj Gupta, "Lecture-05: Recurrent Neural Networks," Siemens Deep Learning and AI series, Nov. 2018

Input Gate

- Selectively updates the cell state based on the new input. A multiplicative input gate unit to protect the stored memory contents from perturbation by irrelevant inputs
  - A sigmoid operation called the "input gate layer" decides the degree of memory update
  - A tanh layer creates a vector of new candidate values that could be added to the memory state.

$$i_t = sigmoid(\theta_{xi} x_t + \theta_{hi} h_{t-1} + b_i)$$

$$\tilde{C}_t = Tanh(\theta_{xg} x_t + \theta_{hg} h_{t-1} + b_g)$$



Adapted from Pankaj Gupta, "Lecture-05: Recurrent Neural Networks," Siemens Deep Learning and AI series, Nov. 2018

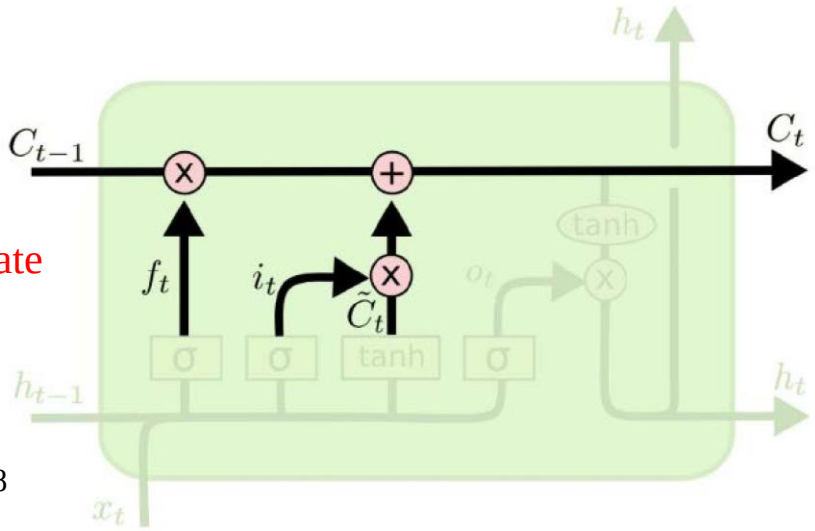
Cell Update

- Update the old cell state, C_{t-1}, into the new cell state C_t
- Multiply the old state by f_t, forgetting the things we decided to forget earlier
- Add $\mathbf{i_t} * \tilde{C}_t$ to get the new candidate values, scaled by how much we decided to update each state value.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

How LSTM tackled vanishing gradient → forget gate

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



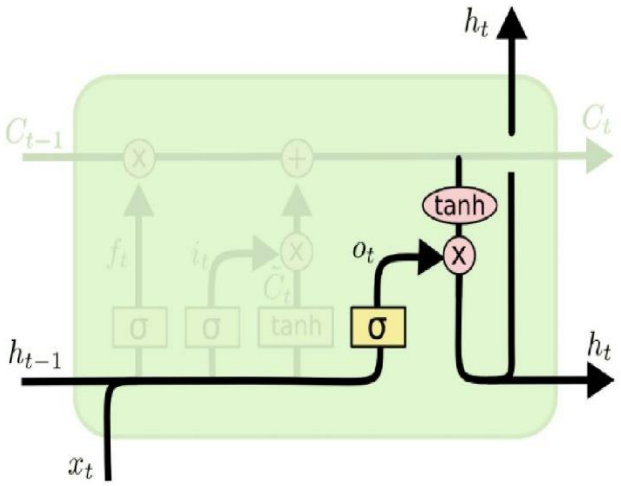
Adapted from Pankaj Gupta, "Lecture-05: Recurrent Neural Networks," Siemens Deep Learning and AI series, Nov. 2018

Output Gate: Filtered Version of the Cell State

- Decides the part of the cell we want as our output in the form of new hidden state
- Multiplicative output gate to protect other units from perturbation by currently irrelevant memory contents
- A sigmoid layer decides what parts of the cell state goes to output.

$$o_t = sigmoid(\theta_{xo} x_t + \theta_{ho} h_{t-1} + b_o)$$

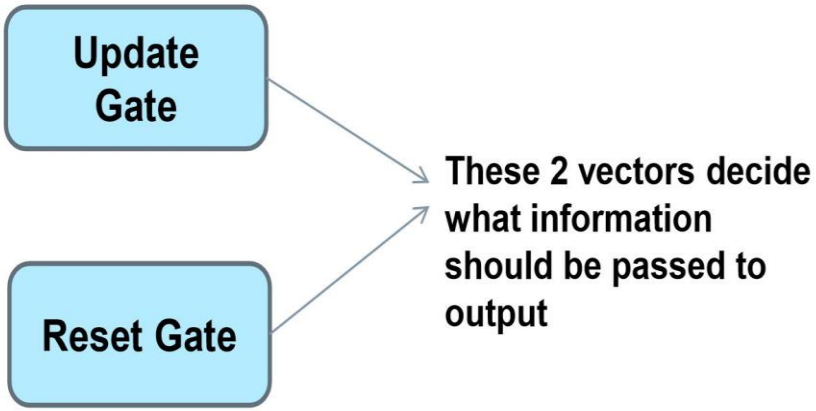
$$h_t = o_t * tanh(C_t)$$

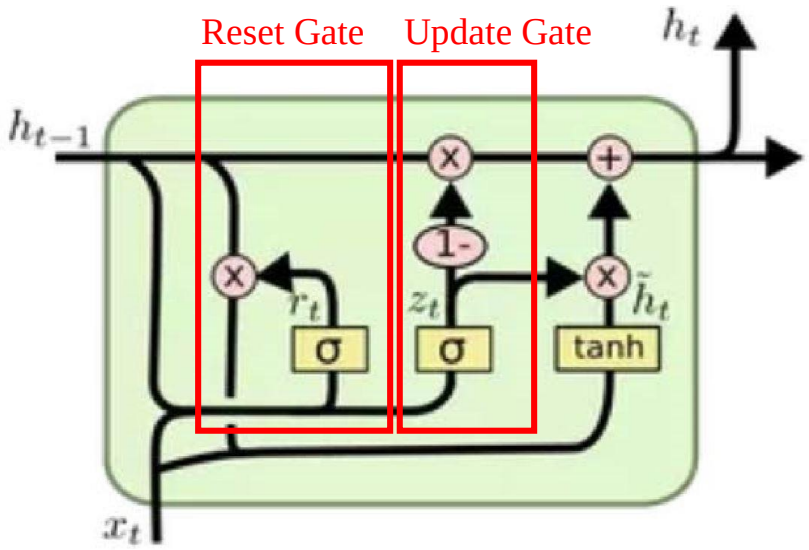


Adapted from Pankaj Gupta, "Lecture-05: Recurrent Neural Networks," Siemens Deep Learning and AI series, Nov. 2018

Gated Recurrent Unit (GRU)

- GRU like LSTM, attempts to solve the vanishing gradient problem in RNN
- Units with short-term dependencies have active reset gates ***r***
- Units with long-term dependencies have active update gates ***z***





Junyoung Chung, et al., "Gated Feedback Recurrent Neural Networks," International Conference on Machine Learning, 2015

Update and Reset Gates

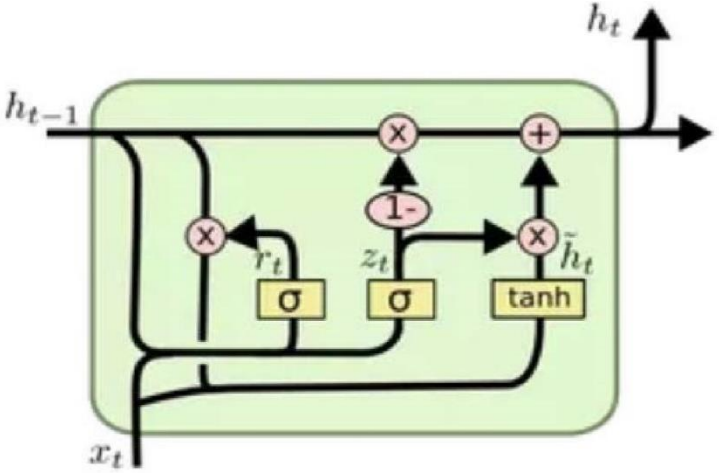
- Update Gate: $z_t = sigmoid(W^z x_t + U^z h_{t-1})$
  - to determine how much of the past information (from previous time steps) needs to be passed along to the future.
  - to learn to copy information from the past such that gradient is not vanished
- Reset Gate: model how much of information to forget by the unit $r_t = sigmoid(W^{(r)} x_t + U^{(r)} h_{t-1})$

Memory Content:

$$h'_t = tanh(W x_t + r\_t \odot U h_{t-1})$$

Final Memory at current time step

$$h_t = z_t \odot h_{(t-1)} + (1 - z_t) \odot h'_t$$

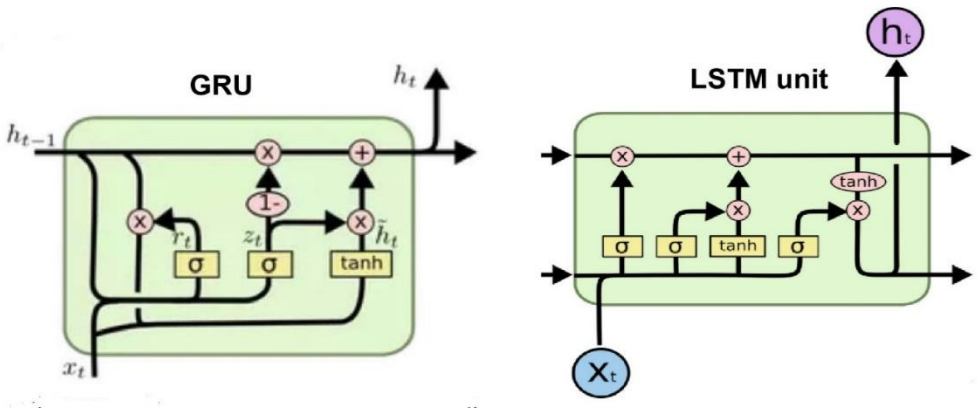


GRU vs. LSTM

- In the LSTM unit, the amount of the cell memory content that is seen, or used by other units in the network is controlled by the output gate. $o_t = sigmoid(\theta_{xo} x_t + \theta_{ho} h_{t-1} + b_o)$

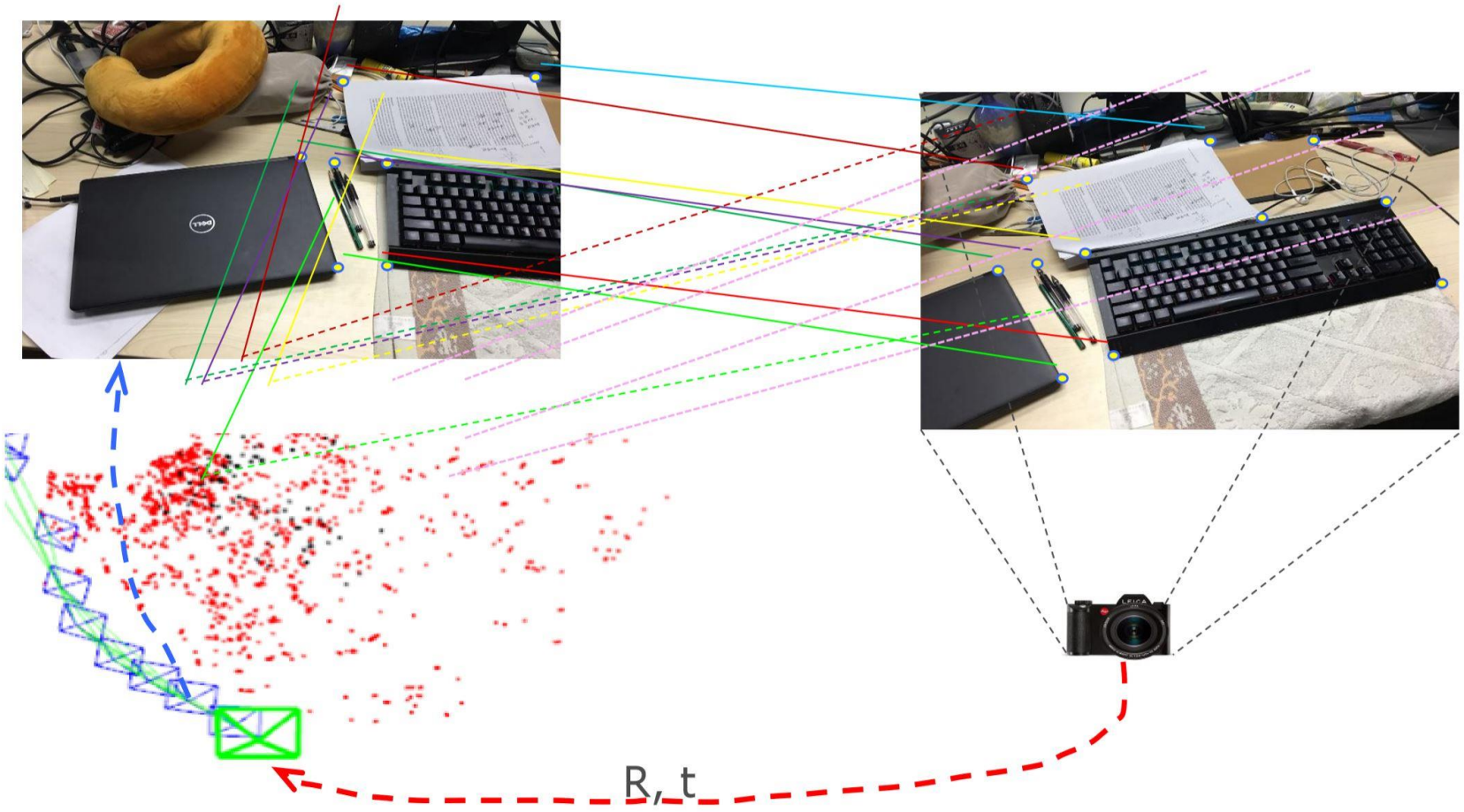
  $h_t = o_t * tanh(C_t)$

- On the other hand, the GRU exposes its full content without any control. GRU performs comparably to LSTM

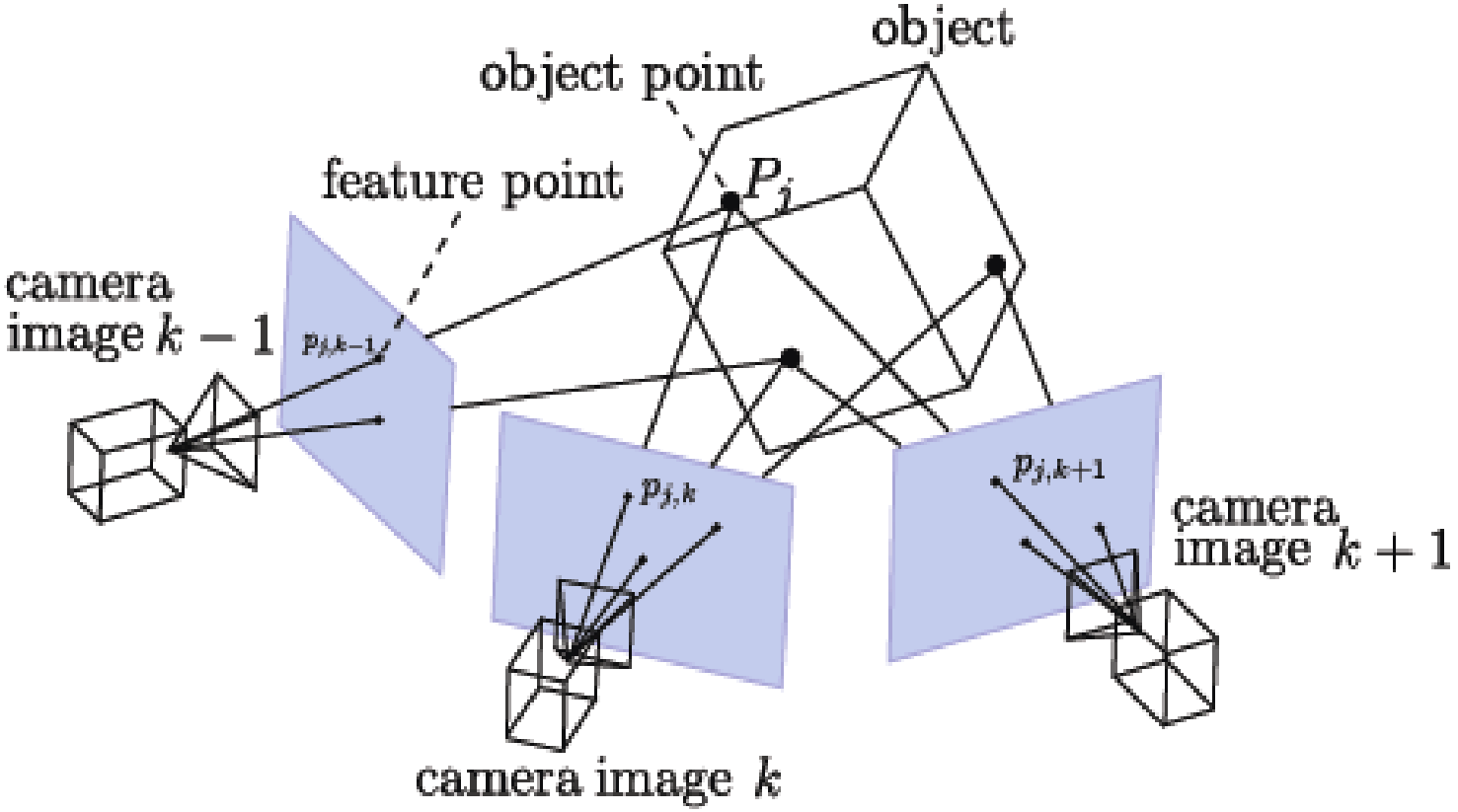


J. Chung et al, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," arXiv 2014

Camera Pose Estimation

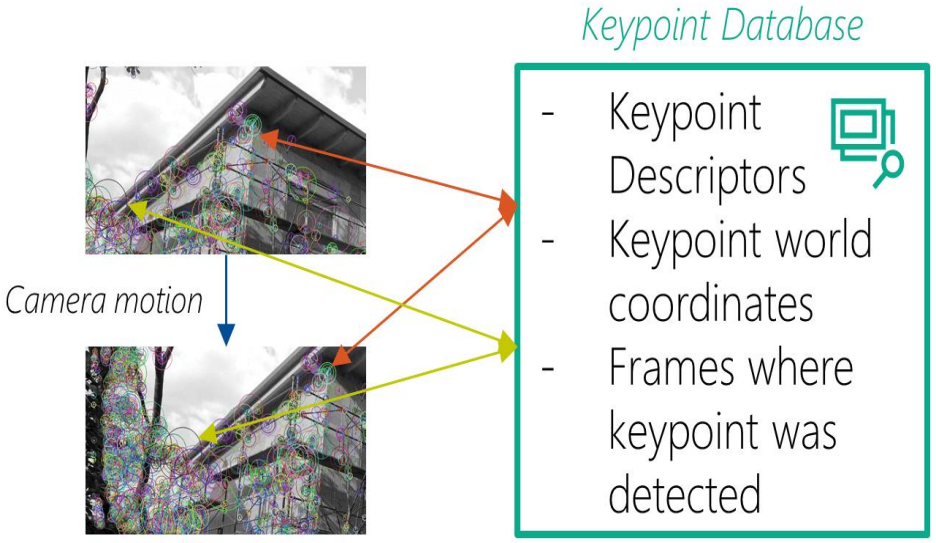


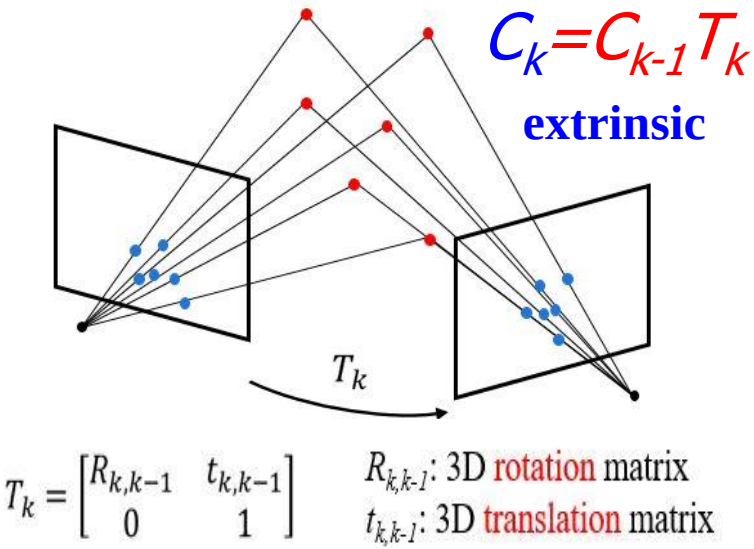
Simultaneously Localization and Mapping (SLAM)



Visual Odometry for Moving Camera Pose Estimation

- Visual Odometry (VO): A SLAM process of determining the position and orientation (extrinsic camera parameters) of a moving object by analyzing the associated camera images.
- **VO** only aims to keep the local consistency of the trajectory



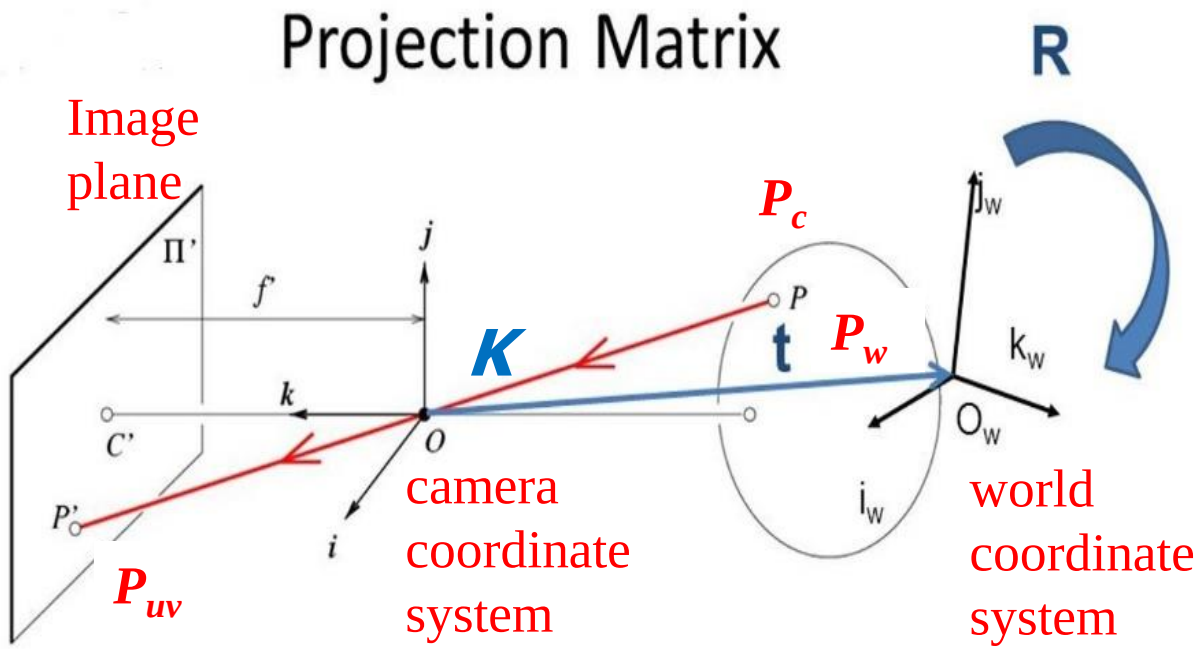


3D-to-2D from Camera Perspective Transform

projection matrix $P_w \rightarrow P_c \rightarrow P_{uv}$

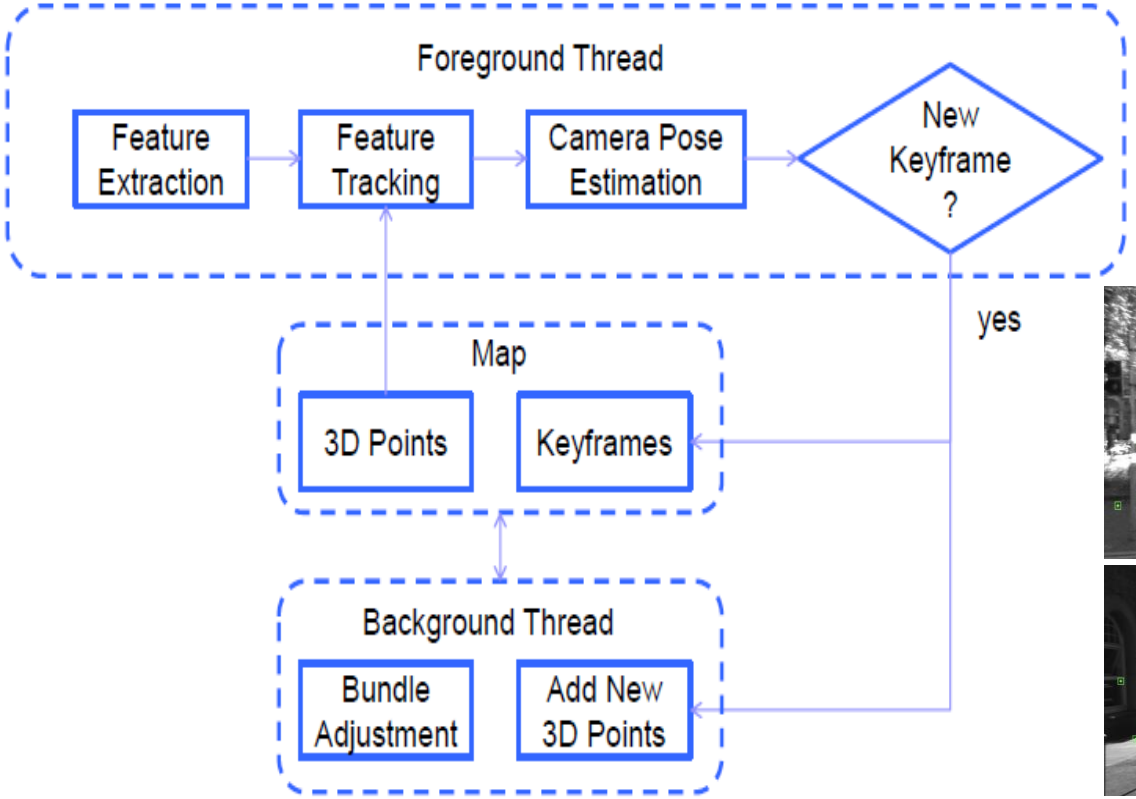
$$[u, v, 1]^T \sim \mathbf{F} \cdot [X, Y, Z, 1]^T \qquad \mathbf{F} = \mathbf{K} \cdot [\mathbf{R}|\mathbf{t}]$$

Intrinsic, Extrinsic

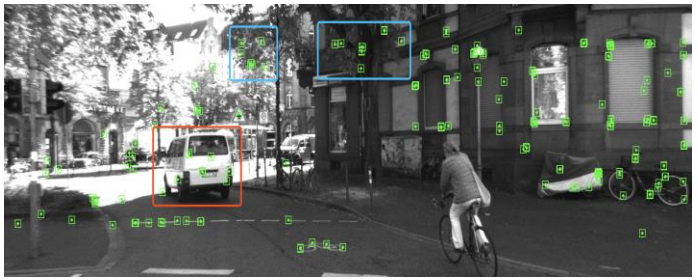


ORB-VO and Sparse Points baed Pose Estimation

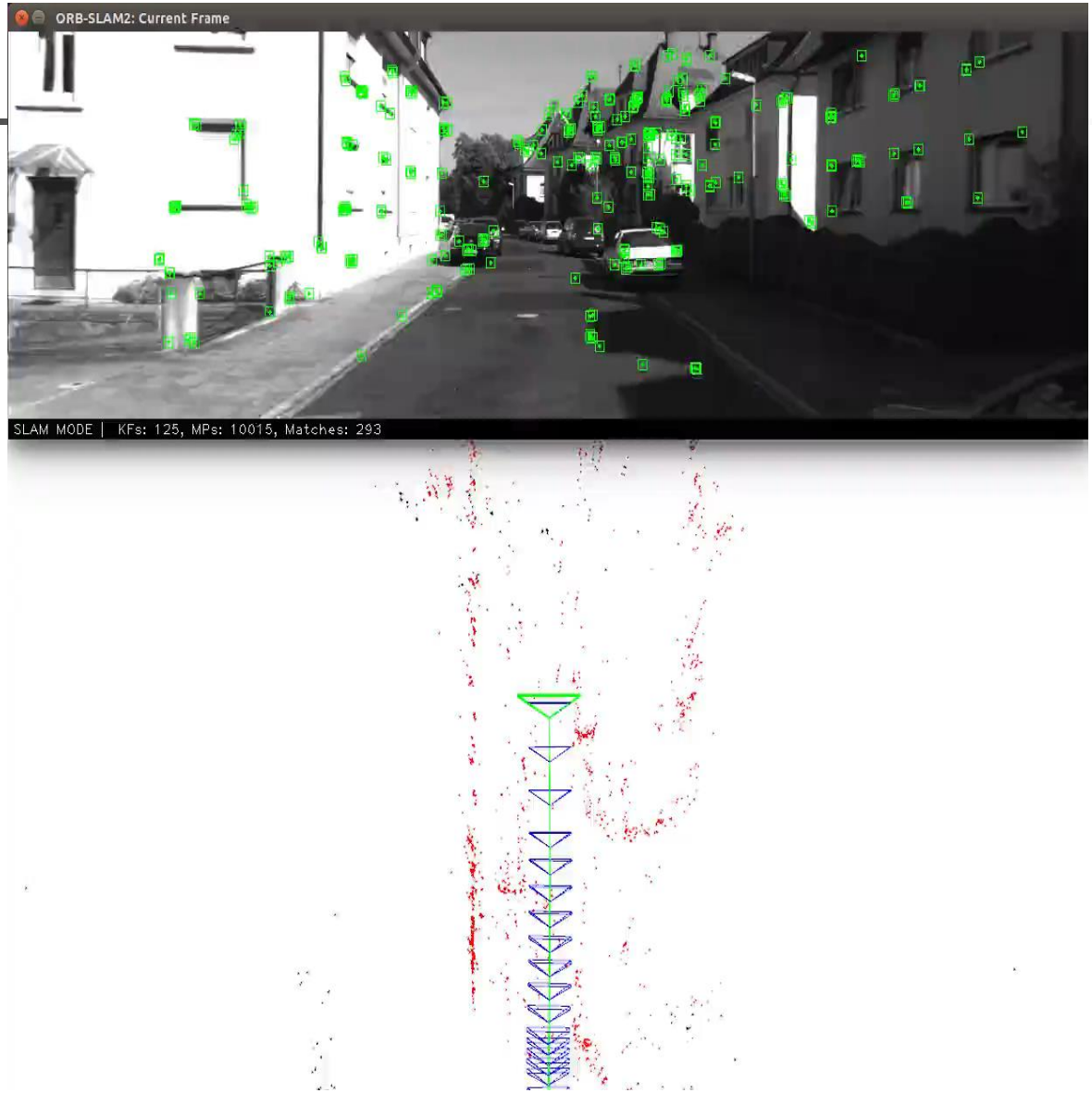
- Camera path computed incrementally (pose after pose)



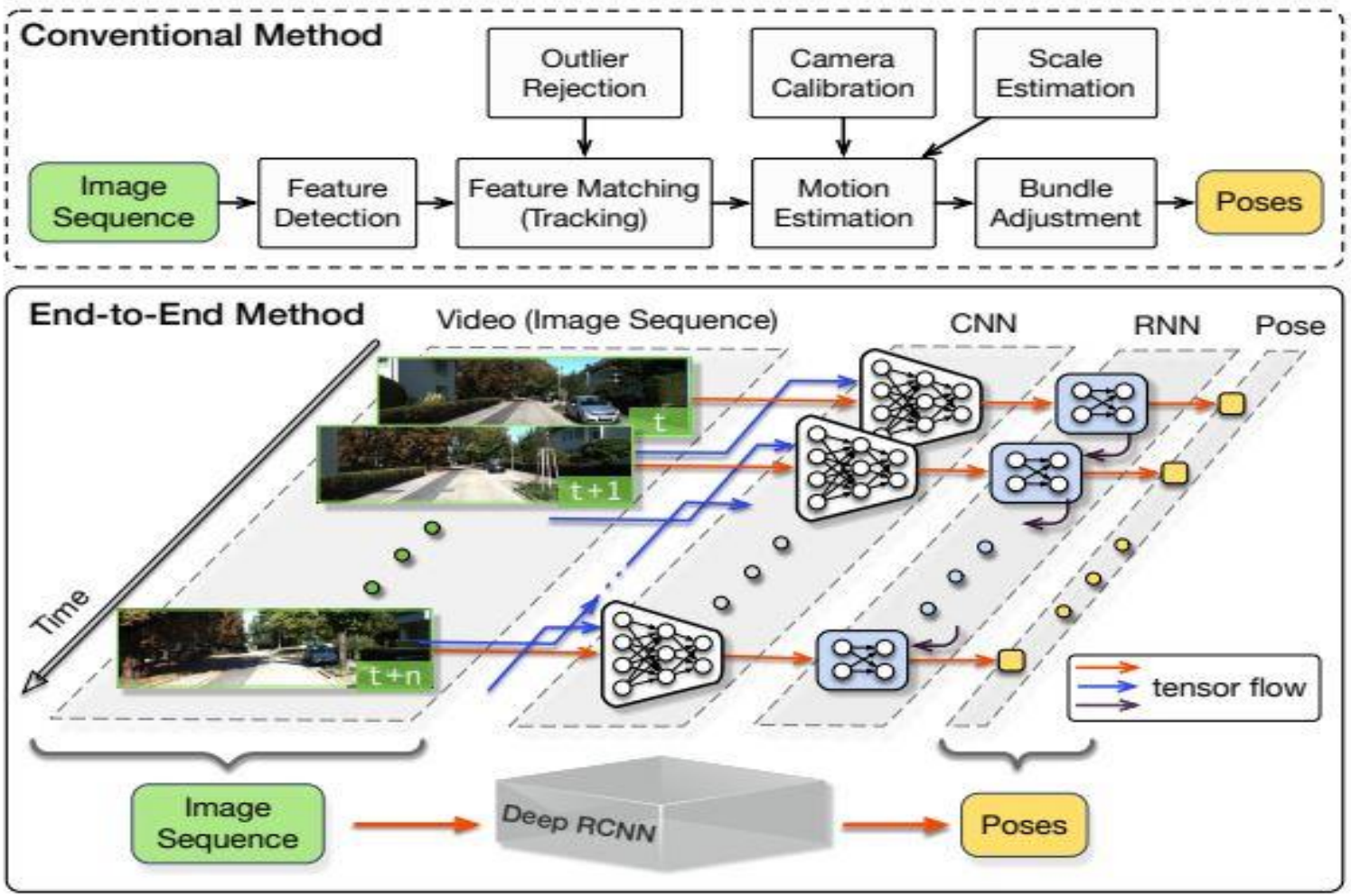
ORB features tracks (oriented FAST and rotated BRIEF)



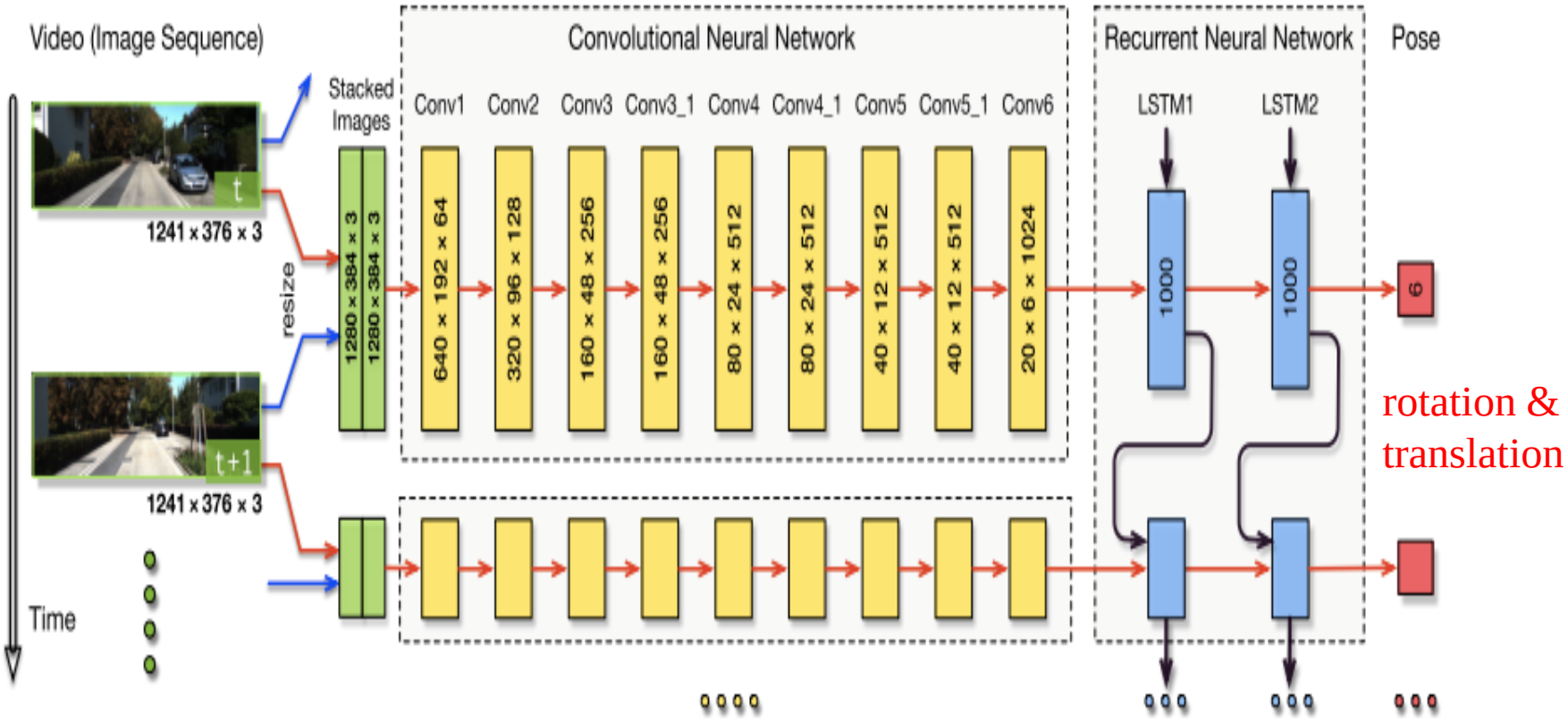
ORB-VO


ORB-SLAM2: Current Frame
SLAM MODE | KFs: 125, MPs: 10015, Matches: 293

CV- vs. Learning-based VO



LSTM for End-to-End VO

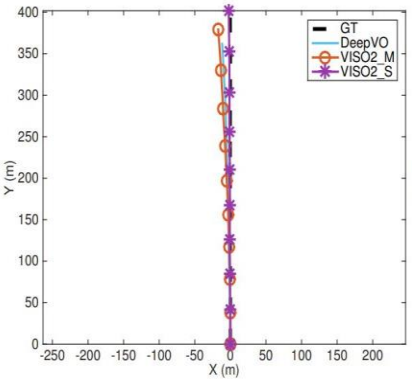


S. Wang, et al., "DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks," IEEE ICRA 2017

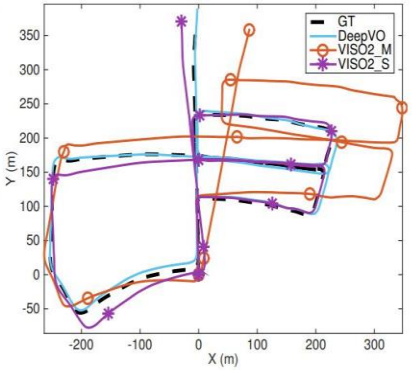
LSTM DeepVO Performance

Seq.	DeepVO		VISO2_M (monocular)		VISO2_S (stereo)	
	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$
03	8.49	6.89	8.47	8.82	3.21	3.25
04	7.19	6.97	4.69	4.49	2.12	2.12
05	2.62	3.61	19.22	17.58	1.53	1.60
06	5.42	5.82	7.30	6.14	1.48	1.58
07	3.91	4.60	23.61	29.11	1.85	1.91
10	8.11	8.83	41.56	32.99	1.17	1.30
mean	5.96	6.12	17.48	16.52	1.89	1.96

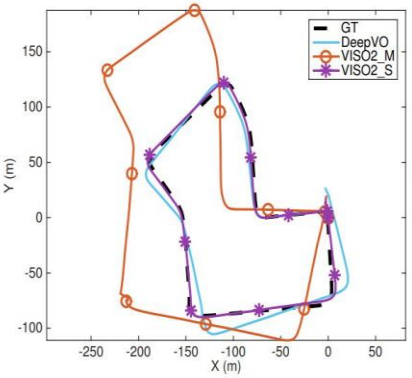
- t_{rel}: average translational RMSE drift $(\%)$ on length of 100m-800m.
- r_{rel}: average rotational RMSE drift $(^\circ/100m)$ on length of 100m-800m.
- The DeepVO model used is trained on Sequence 00, 02, 08 and 09. Its performance is expected to improve when it is trained on more data.



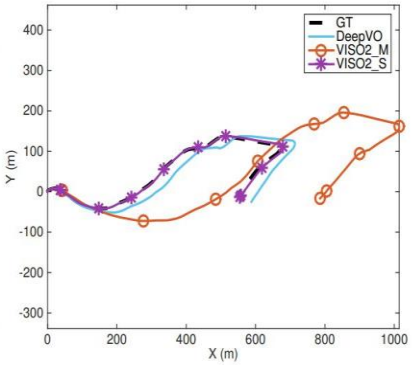
(a) Sequence 04.



(b) Sequence 05.



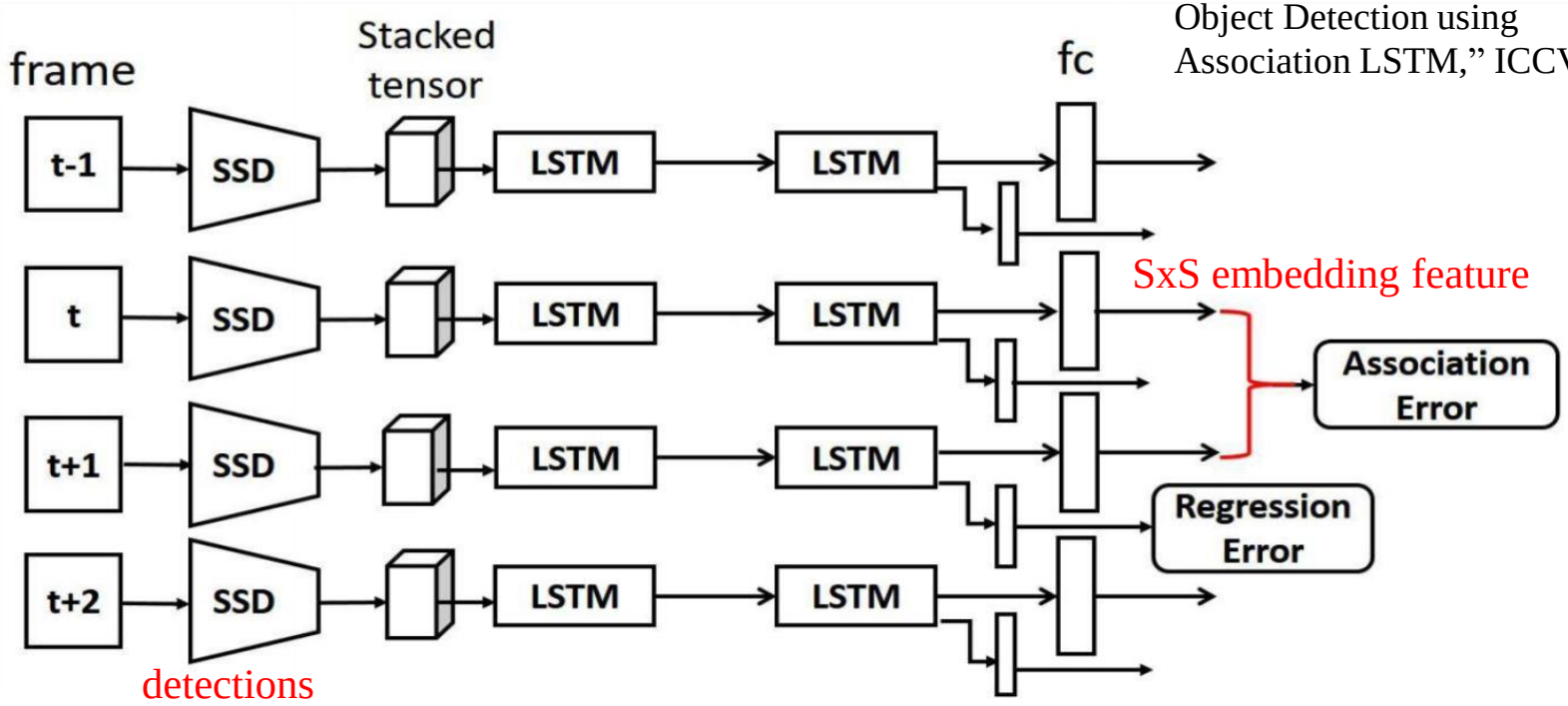
(c) Sequence 07.



(d) Sequence 10.

LSTM for Detection via Temporal Association in Videos

Yongyi Lu, et al., "Online Video Object Detection using Association LSTM," ICCV 2017



- Solve the regression and association tasks in a unified framework → detection-by-tracking

Smoothed Detections

temporal continuity

$$\mathcal{L}_{reg}(l, g, c) = \sum (L_{conf}(c, c^*) + \lambda L_{loc}(l, g)) + \alpha \cdot \mathcal{L}_{smooth}$$

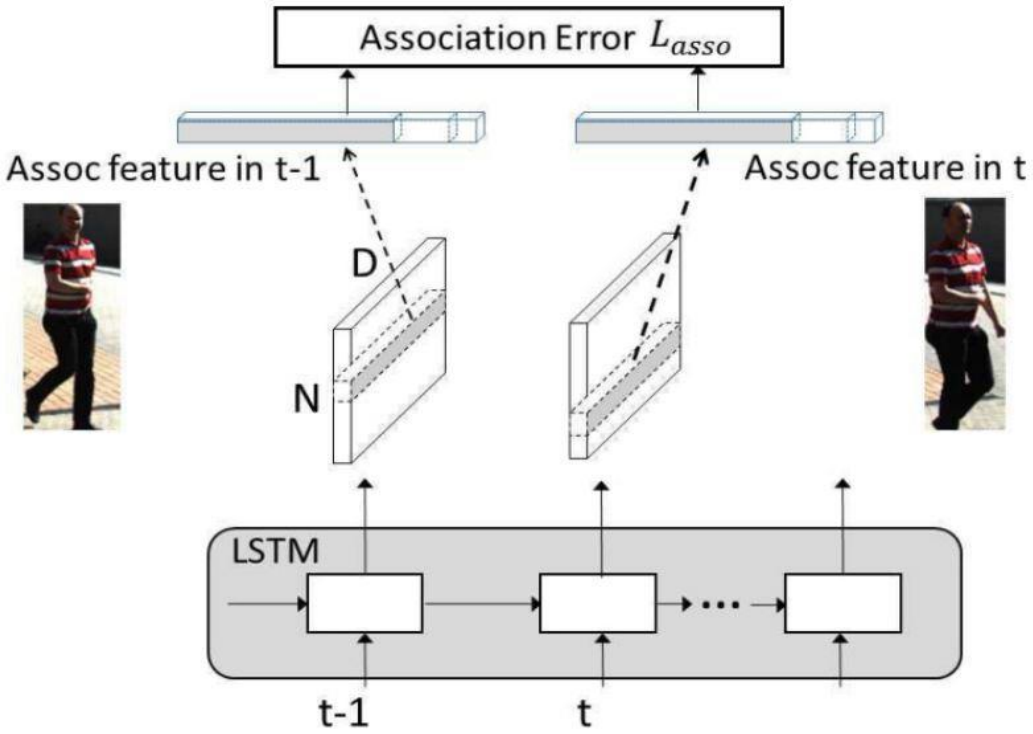
$$\mathcal{L}_{smooth} = \sum_{\tau} (\tilde{l}_t - \tilde{l}_{t+1})$$

- The first two terms are the object regression error, where the localization loss L_{loc} is a smooth L_1 loss between the predicted box (l) and ground truth box (g).
- The confidence loss L_{conf} is the softmax loss over multiple classes confidences c toward ground truth score vector c^*.
- The LSTM model is regularized by applying the smoothness constraint across consecutive time-steps.

Feature Associations

$$\mathcal{L}_{asso} = \sum_{t} \sum_{i,j} \theta_{ji} |\phi_{t-1}^{i} \cdot \phi_{t}^{j}|$$

$$\theta_{jk} \in 0, 1$$



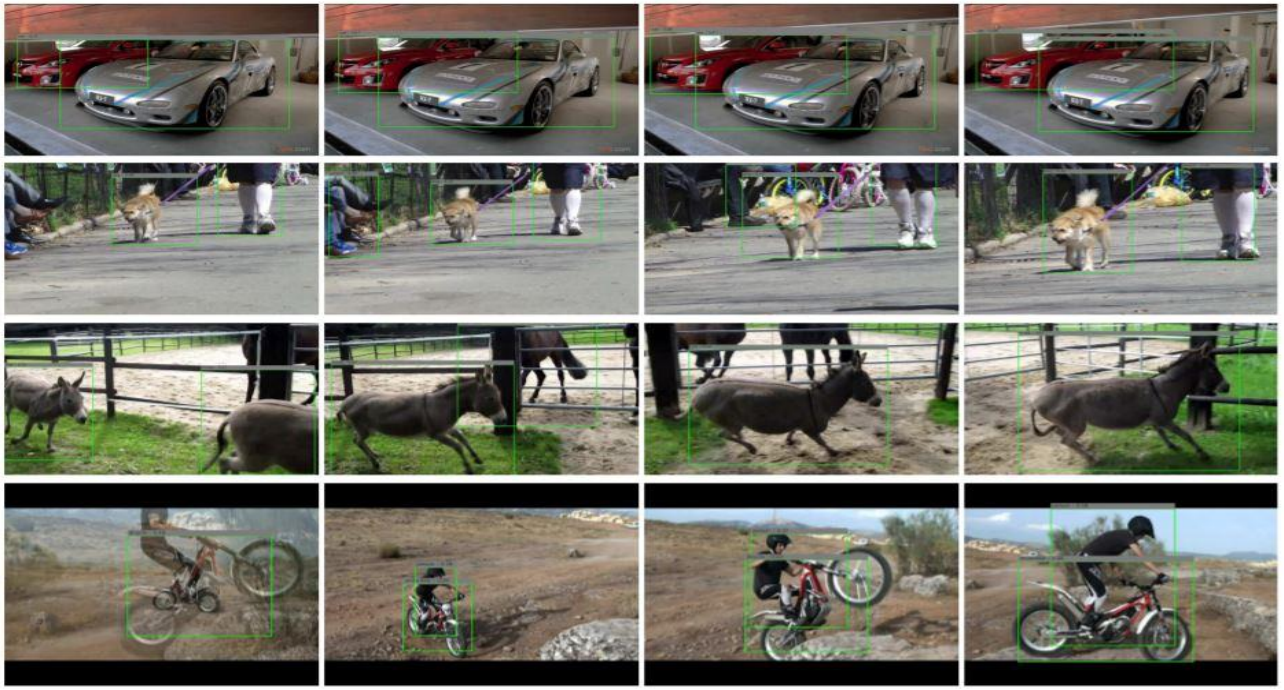
ϕ_t^j is the *j*-th detected objected association feature in frame *t*.

- *D (=c+4+ SxS)* is the composite feature length for each detected object
- *N* is the top number of detected objects.

Detection Performance

Methods	mAP	airplane	bird	boat	car	cat	cow	dog	horse	mbike	train
VOP [25]	37.41	29.77	28.82	35.34	41.00	33.7	57.56	34.42	54.52	29.77	29.23
Unsupervised [12]	55.7	56.5	66.4	58.0	76.8	39.9	69.3	50.4	56.3	53.0	31.0
YOLO [21]	56.53	76.67	89.51	57.66	65.52	43.03	53.48	55.81	36.96	24.62	62.03
Context [27]	68.73	76.11	87.65	62.16	80.69	62.42	78.02	58.72	81.77	41.54	58.23
Baseline_1	66.21	74.89	85.03	60.11	77.63	61.22	77.56	56.91	80.18	40.67	54.83
Baseline_2	70.43	77.14	91.02	63.34	81.70	63.47	79.38	59.18	83.56	42.33	59.80
a_LSTM	**72.14**	78.92	90.94	65.87	84.76	65.22	81.39	61.86	83.27	43.92	61.25

Table 1. Per-category object detection results on the Youtube-Objects datasets. *a_LSTM* denotes our proposed association LSTM.



MoTChallenge 2015 Results (Poor in Tracking)

Methods	MOTA ↑	MOTP ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDS ↓
MDP (K=5) [28]	26.7	73.7	12.0	53.0	3,386	13,415	111
MDP (K=9) [28]	26.7	73.6	12.0	51.7	3,290	13,491	133
CDT [11]	39.9	74.8	20.9	47.4	914	12,856	95
a_LSTM	38.6	74.2	14.9	**46.8**	**788**	13,253	154

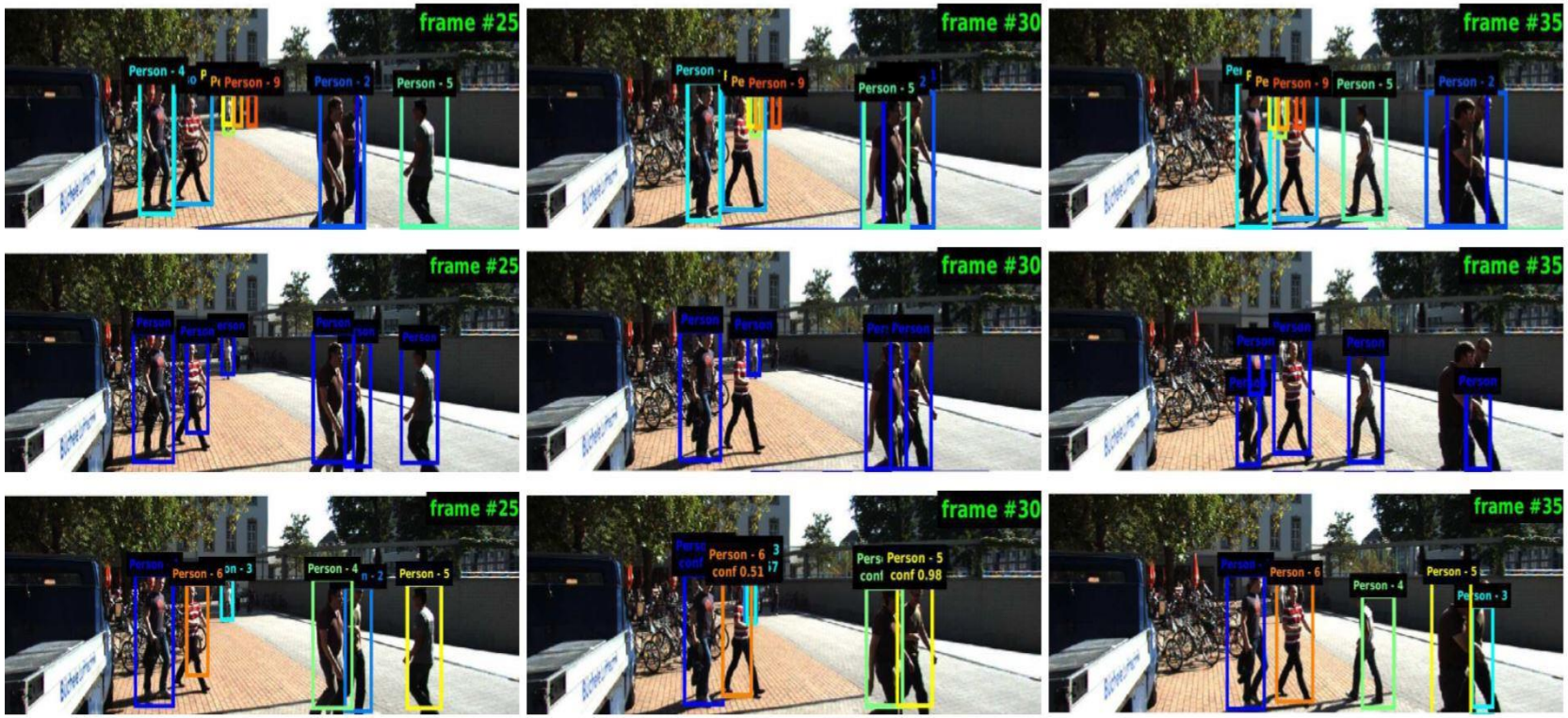
SOTA

Tracker	↑MOTA	IDF1	HOTA	MT	ML	FP	FN	RcII	Prcn	AssA	DetA	AssRe	AssPr	DetRe	DetPr	LocA	FAF	ID Sw.	Frag	Hz
GMOTv2 1.	**64.5**±12.0	**67.0**	**51.1**	**389** (54.0)	112 (15.5)	9,613	**11,872**	**80.7**	83.8	**49.1**	**53.5**	**58.1**	67.3	**63.6**	66.0	79.4	1.7	315 (0.0)	808 (0.0)	8.6
NEUT 2.	55.1±10.7	61.1	46.2	229 (31.8)	129 (17.9)	5,532	21,582	64.9	87.8	46.3	46.4	51.1	75.4	52.1	70.5	80.9	1.0	484 (0.0)	2,248 (0.0)	21.8
STTA 3.	53.4±12.3	59.8	45.3	212 (29.4)	203 (28.2)	6,774	21,204	65.5	85.6	46.0	45.1	52.1	70.0	51.4	67.1	79.2	1.2	628 (0.0)	1,090 (0.0)	3.1
Lif_T 4.	52.5±0.0	60.0	46.0	244 (33.8)	186 (25.8)	6,837	21,610	64.8	85.3	47.8	44.9	53.7	75.0	51.2	67.4	79.8	1.2	730 (11.3)	1,047 (16.2)	1.5
	A. Hornakova, R. Henschel, B. Rosenhahn, P. Swoboda. Lifted Disjoint Paths with Application in Multiple Object Tracking. In ICML, 2020.																			
MPNTrack 5.	51.5±10.8	58.6	45.0	225 (31.2)	187 (25.9)	7,620	21,780	64.6	83.9	46.2	44.4	54.8	67.1	51.0	66.3	79.4	1.3	375 (5.8)	872 (13.5)	6.5
	G. Brasó, L. Leal-Taixé. Learning a Neural Solver for Multiple Object Tracking. In CVPR, 2020.																			
ApLift 6.	51.1±13.9	59.0	45.7	284 (39.4)	163 (22.6)	10,070	19,288	68.6	80.7	46.7	45.4	55.2	68.7	54.1	63.7	79.3	1.7	677 (0.0)	1,022 (0.0)	0.6
	A. Hornakova*, T. Kaiser*, M. Rolinek, B. Rosenhahn, P. Swoboda, R. Henschel. Making Higher Order MOT Scalable: An Efficient Approximate Solver for Lifted Disjoint Paths. In International Conference on Computer Vision (ICCV), 2021.																			
ITM 7.	50.1±10.6	59.8	40.2	286 (39.7)	**95** (13.2)	7,638	21,311	65.3	84.0	37.0	44.4	42.4	68.7	51.4	66.1	79.6	1.3	1,730 (26.5)	1,891 (29.0)	1.2
mfi_tst 8.	49.2±10.7	62.4	41.5	210 (29.1)	176 (24.4)	8,707	21,594	64.9	82.1	40.3	43.4	44.2	74.2	50.7	64.2	79.0	1.5	912 (0.0)	1,397 (0.0)	0.7
	J. Y, H. Ge, J. Yang, Y. Tong, S. Su. Online multi-object tracking using multi-function integration and tracking simulation training. In Applied Intelligence, 2021.																			
FGRNetIV 9.	47.5±9.7	47.8	0.0	169 (23.4)	190 (26.4)	5,531	25,502	58.5	86.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1,240 (21.2)	1,550 (26.5)	3.2
Lif_TsimInt 10.	47.2±15.8	57.6	43.8	195 (27.0)	215 (29.8)	7,635	24,277	60.5	83.0	46.7	41.5	52.4	74.5	47.8	65.5	79.4	1.3	554 (9.2)	803 (13.3)	5.8
	A. Hornakova, R. Henschel, B. Rosenhahn, P. Swoboda. Lifted Disjoint Paths with Application in Multiple Object Tracking. In ICML, 2020.																			

Metric	Definition
MOTA ↑	It considers false positives, false negatives and ID-Switches.
MOTP ↑	It measures the tightness of the tracking results and ground truth.
MT ↑	More than 80% of tracks are successfully tracked.
ML ↓	Less than 20% of tracks are successfully tracked.
FP ↓	The total number of false positives.
FN ↓	The total number of false negatives (missed targets).
IDS ↓	Total number of times that an output track changes its identity.

Tracking Performance

- (1st row) ground truth. (2nd row) object detection responses (3rd row) occlusion, moving camera and change of scale.

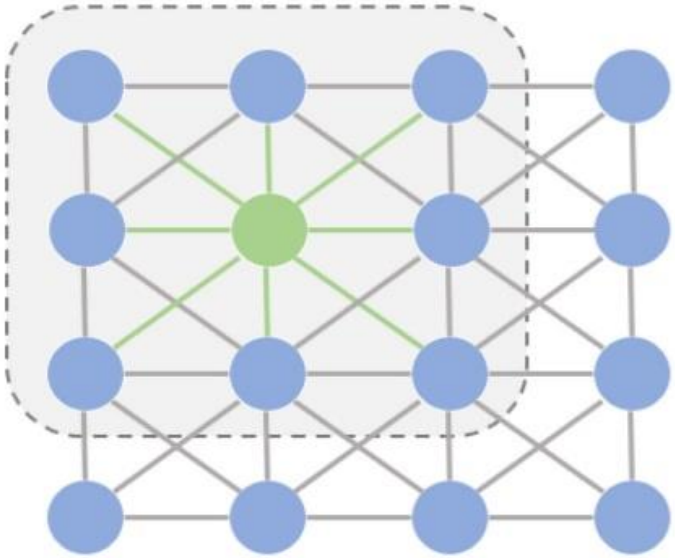


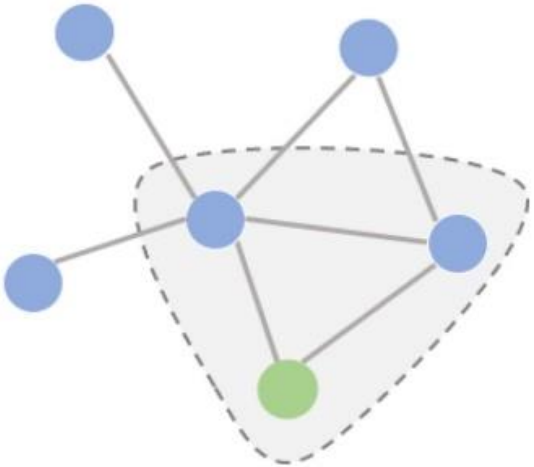
Graph Convolution Network (GCN) & Graph R-CNN

Image vs. Graph

- Images are defined over Euler space, as indexed by (i,j) grid points, with a fixed neighborhood on each pixel, which allow convolution, but NOT graph

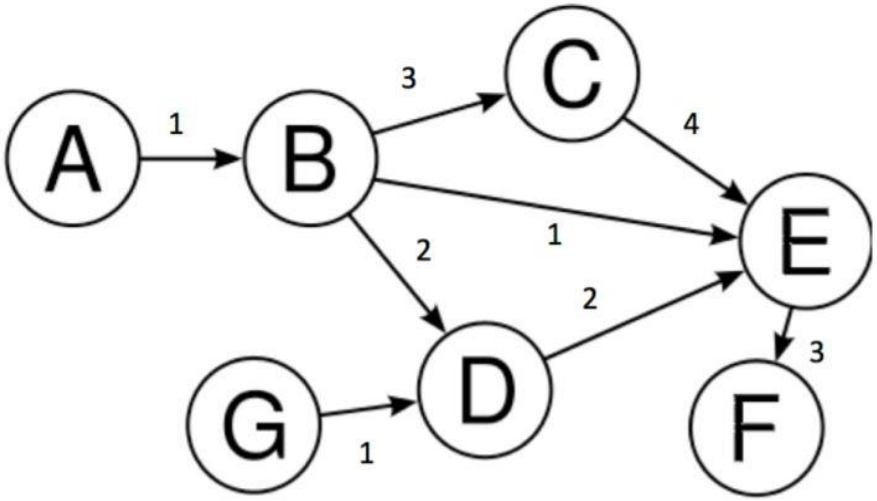
$$y(i,j) = \sum_{m,n} h(m,n)x(i-m, j-n)$$





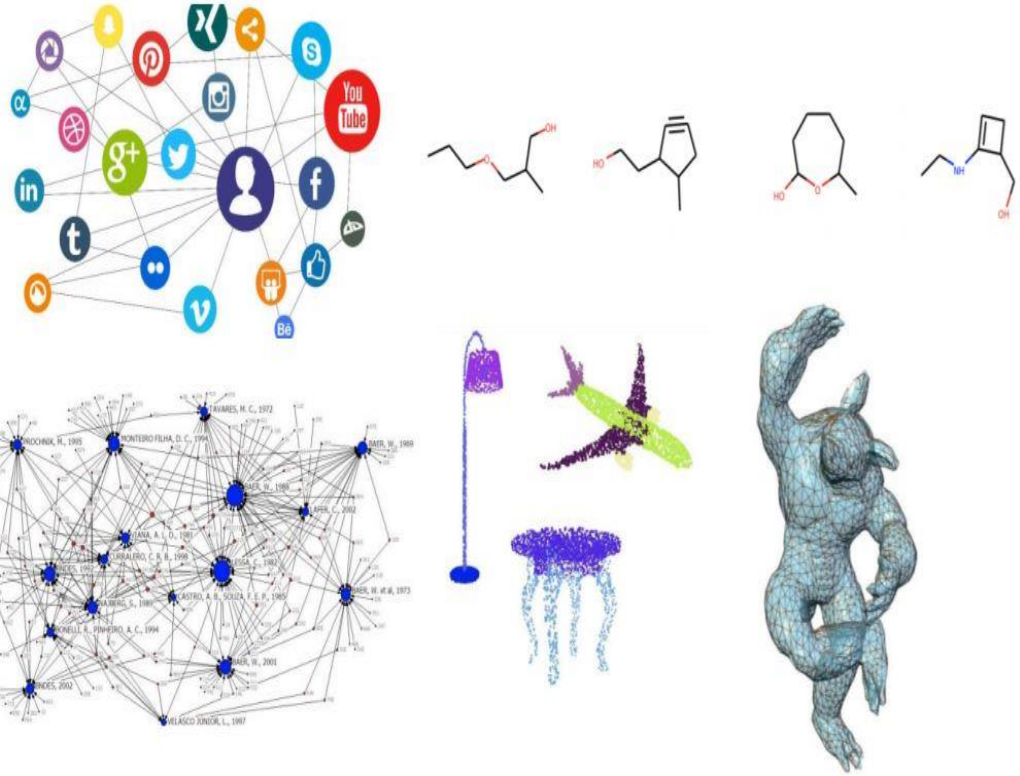
A Graph

- A graph (directed or undirected) consists of a set of vertices V (or nodes) and a set of edges E
- Edges can be weighted (weights can be scalar or vector) or binary
- Nodes are represented by attribute values (can be scalar or vector)



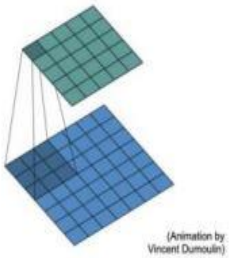
Why Graph Convolution Networks?

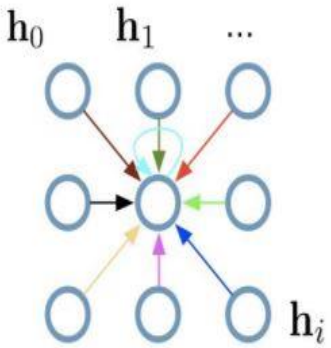
- Lots of real-world applications need to deal with Non-Grid data
  - General Graphs
  - Social Networks
  - Citation Networks
  - Molecules
  - Point Clouds
  - 3D Meshes
  - ...



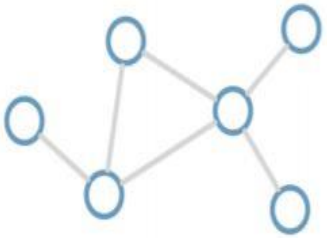
CNN vs GCN

Single CNN layer with 3x3 filter:

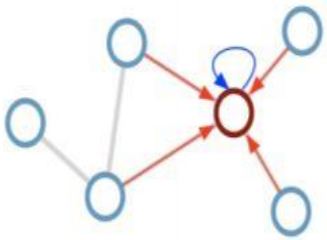




Consider this undirected graph:



Calculate update for node in red:



Full update:

$$h_4^{(l+1)} = \sigma\left(W_0^{(l)} h_0^{(l)} + W_1^{(l)} h_1^{(l)} + \cdots + W_8^{(l)} h_8^{(l)}\right)$$

Update rule:

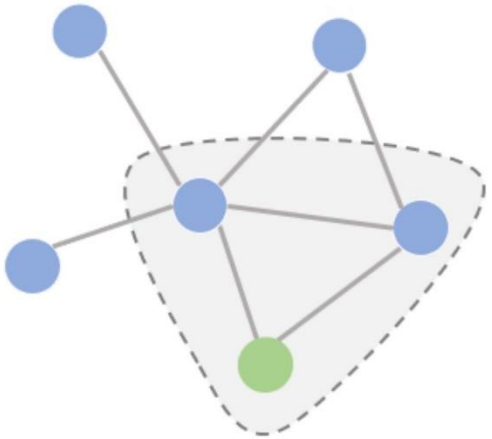
$$h_i^{(l+1)} = \sigma\left(h_i^{(l)} W_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} h_j^{(l)} W_1^{(l)}\right)$$

Convolutional Neural Network (CNN)

Graph Convolutional Network (GCN)

$\mathcal{N}_i$: neighbor indices c_{ij} : norm. constant (fixed/trainable)

By Thomas Kipf.



GCN Definition

- To learn a function of signals/features on a graph $G=(V,E)$ which takes as input:
  - A feature description x_i for every node i; summarized in an $N \times D$ feature matrix X (N: number of nodes, D: dimension of input features)
  - A representative description of the graph structure in matrix form; typically in the form of an adjacency matrix A (or some function thereof)
- and produces a node-level output Z (an $N \times F$ feature matrix, where F is the dimension of output features per node).

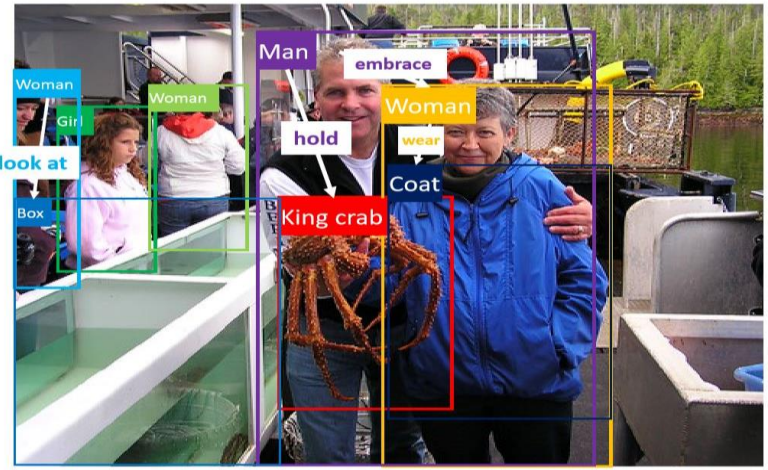
$$H^{(l+1)} = f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right) \quad \text{with } H^{(0)} = X \text{ and } H^{(L)} = Z$$

- where $W^{(l)}$ is a weight matrix for the l-th neural network layer and $\sigma(\cdot)$ is a non-linear activation function like the ReLU.

Human Object Interactions via GCN

"Woman look at box"

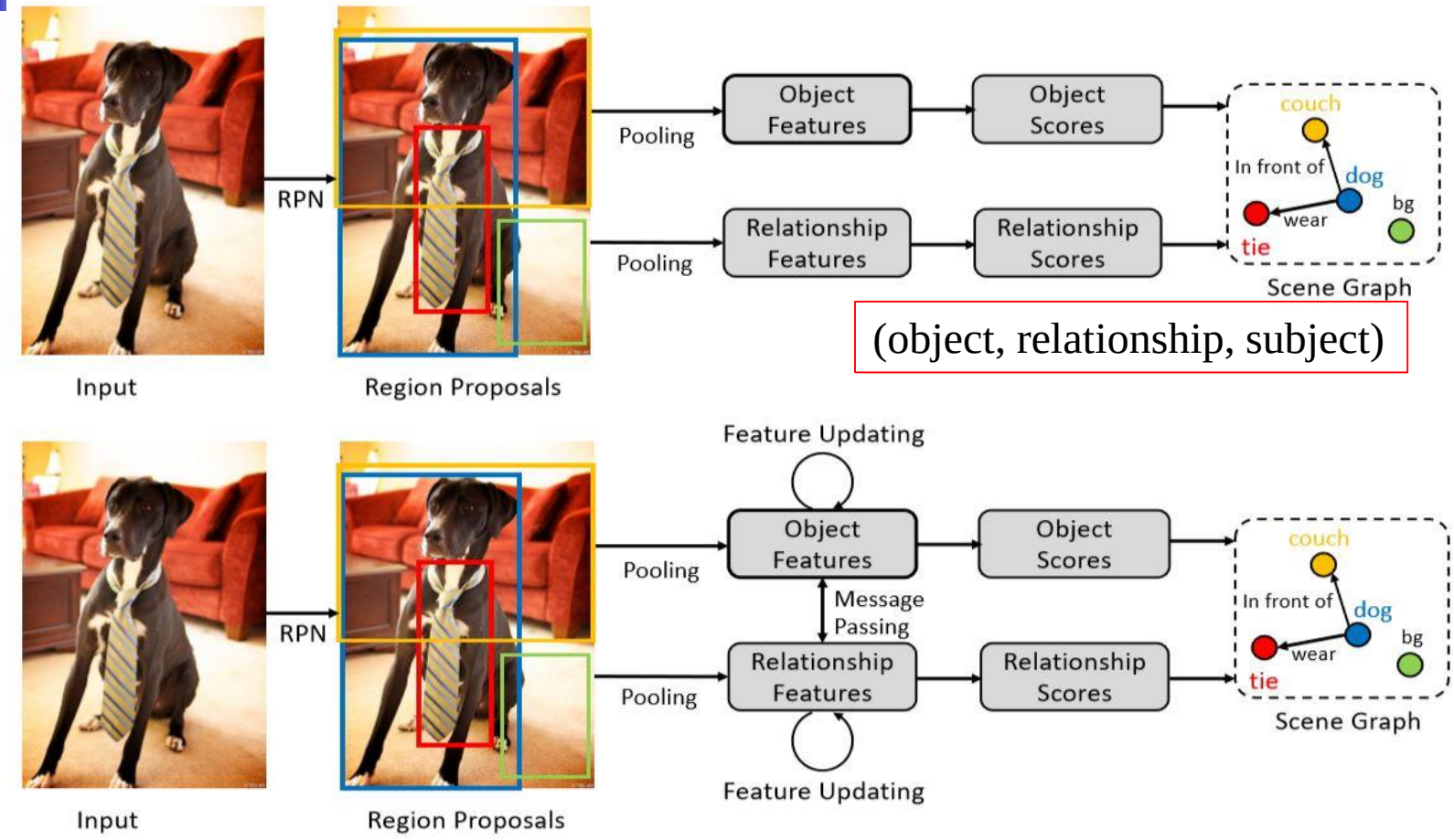
"Man hold king crab"



"Woman wear coat"

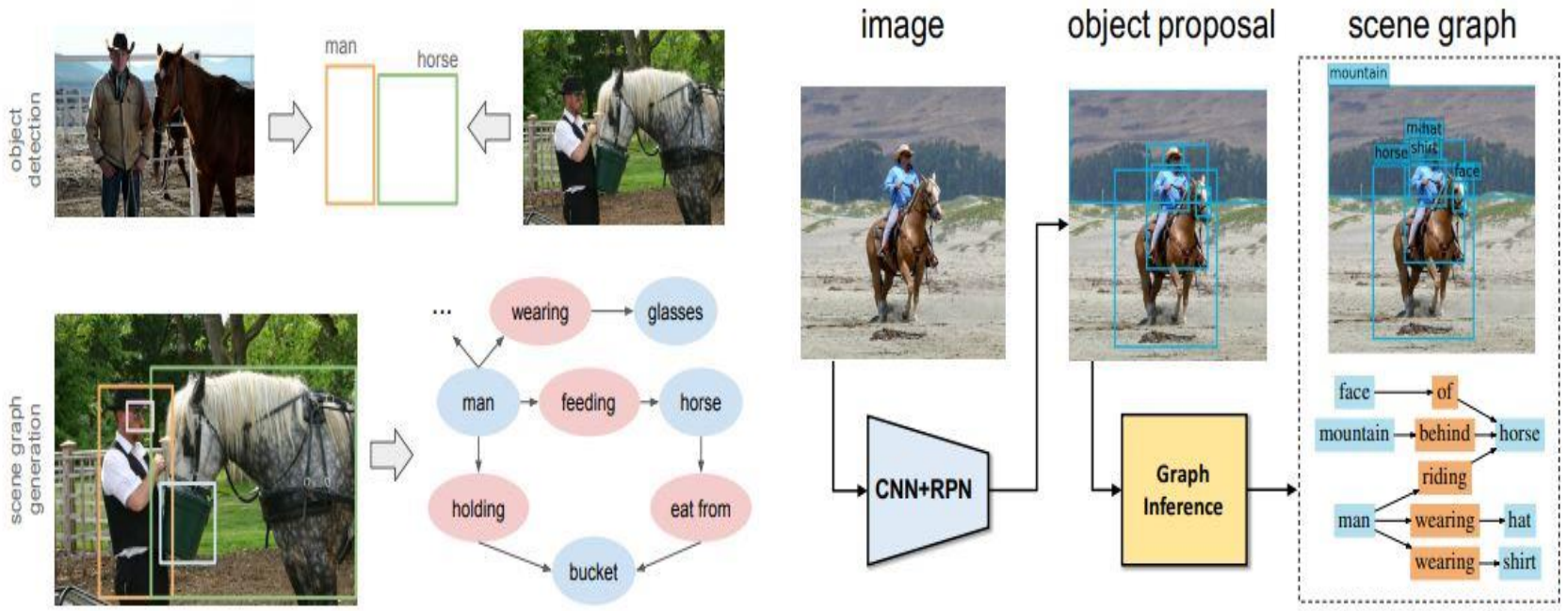
"Man embrace woman"

A Base Model for Scene Graph Generation



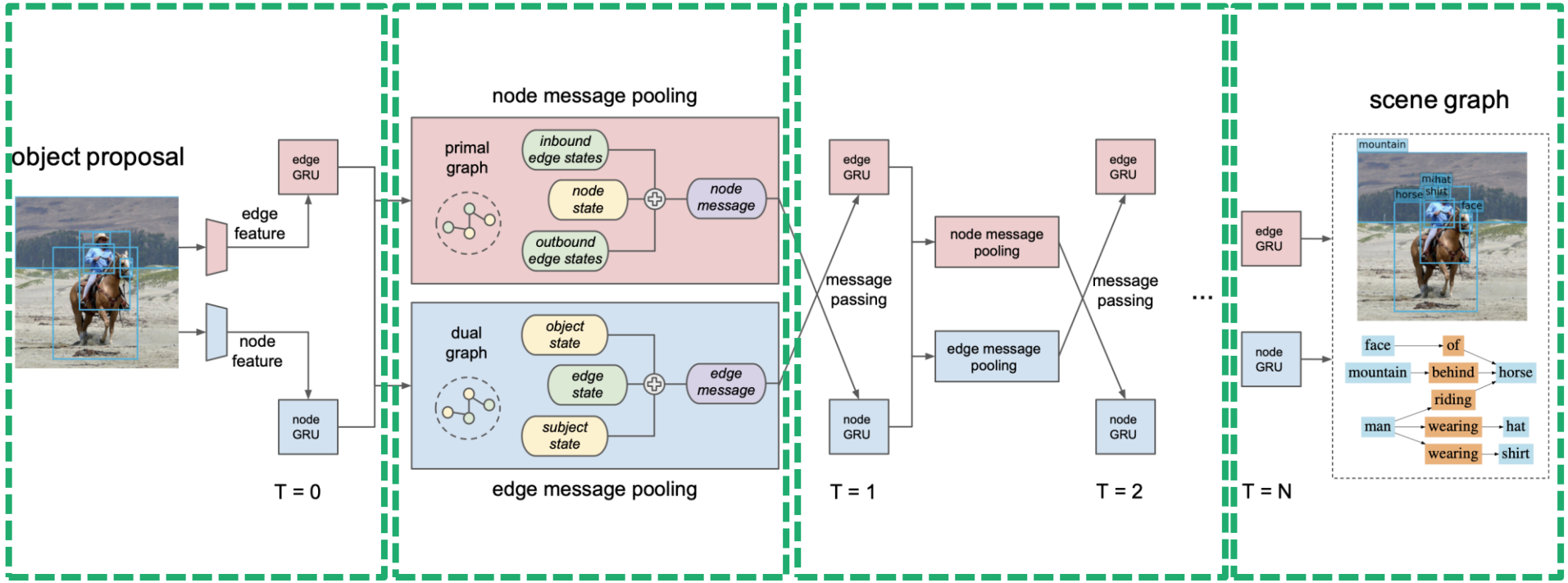
Danfei Xu, et. al, "Scene Graph Generation by Iterative Message Passing (IMP)," CVPR 2017

Scene Graph Building

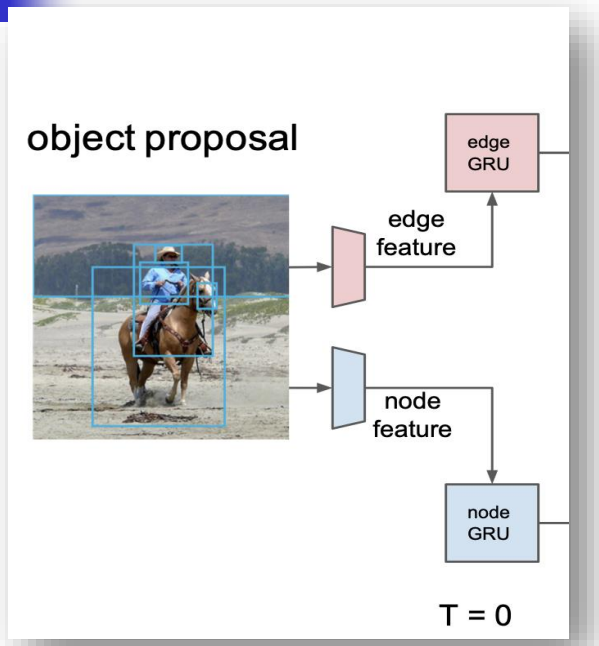


- The model first produces a set of object proposals using a Region Proposal Network (RPN)
- Pass the extracted features of the object regions to a graph inference module.
- The output of the model is a scene graph, which contains a set of localized objects (bbox regression), categories of each object (classification), and relationship types between "each pair of objects".

IMP Network Step by Step



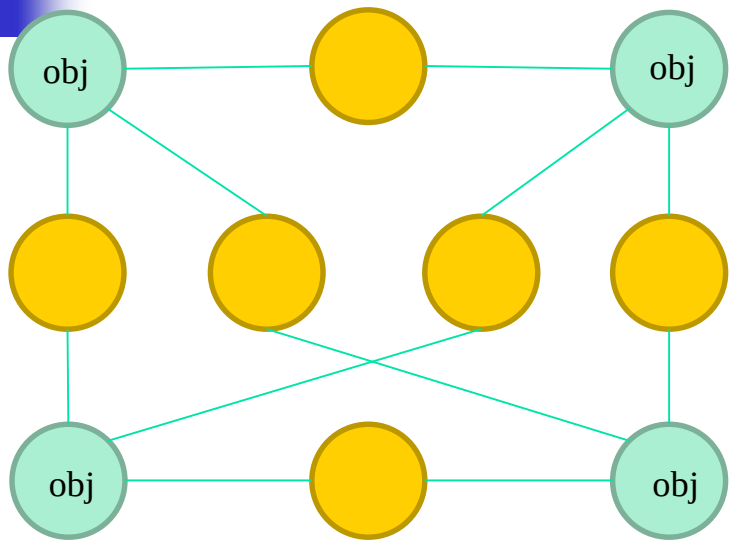
Step 1: Object Proposal



- T = 0
- Generate visual features ***v*** for object proposals (VGG+RPN)
- GRUs take ***v*** as input, and generate hidden states

- For the RPN
  - CNN weights are fixed
  - FC to fine-tuned
  - (NMS and sample 128 for training, 50 for testing)

Step 1: IMP Graph



"*Simplified*" example graph for 4 objects

Node GRU for object

Edge GRU for relationship

The ⊕ symbol denotes a learned weighted sum

Connection (message passing)

visual feature

"girl"

Union box

"horse"

Object 1

Object 2

graph nodes

"riding"

message

message

Object 1

relationship

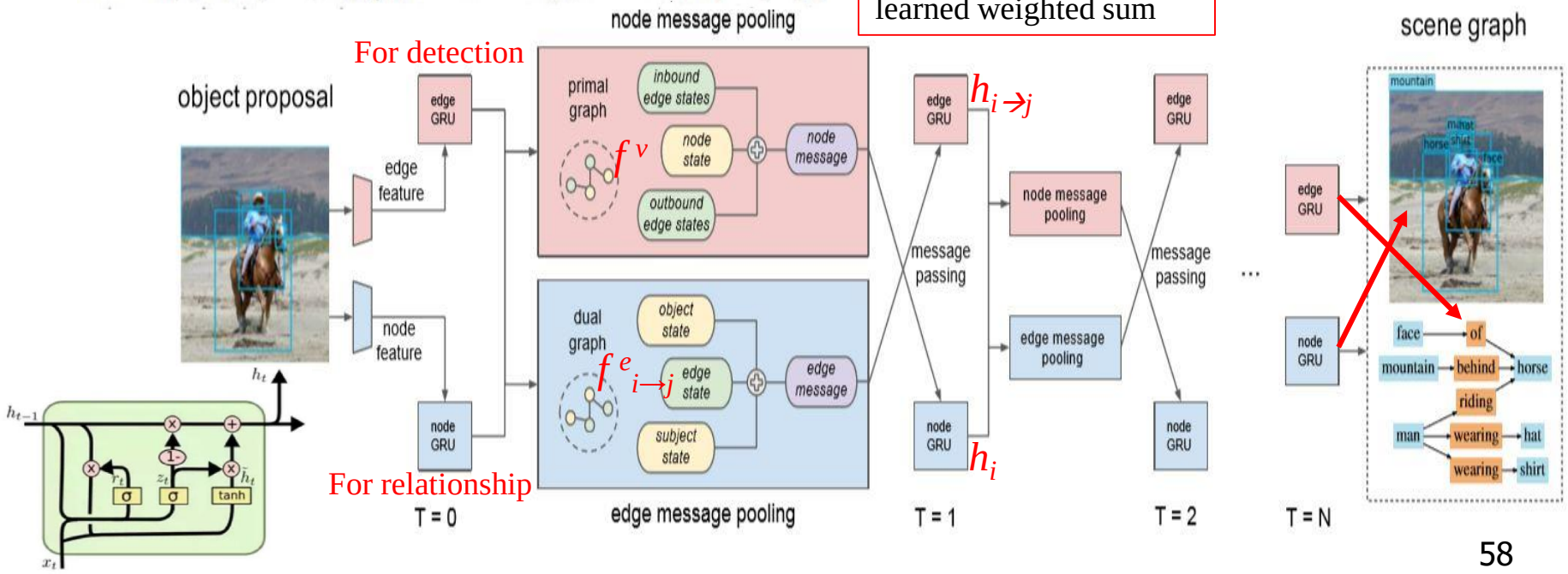
Object 2

IMP for Scene Graph

- The visual feature f^v (512-dim) of the 128 proposal box is used as feature for each node.
- The visual feature of the union box over the proposal boxes (b_i,b_j) as the visual feature
- $f^e_{i\to j}$ (512-dim) for edge $i \to j$. These visual features are extracted by an ROI pooling layer from the image.

$$\mathbf{x} = \{x_i^{cls}, x_i^{bbox}, x_{i\to j} | i = 1 \ldots n, j = 1 \ldots n, i \neq j\}$$

The $\oplus$ symbol denotes a learned weighted sum

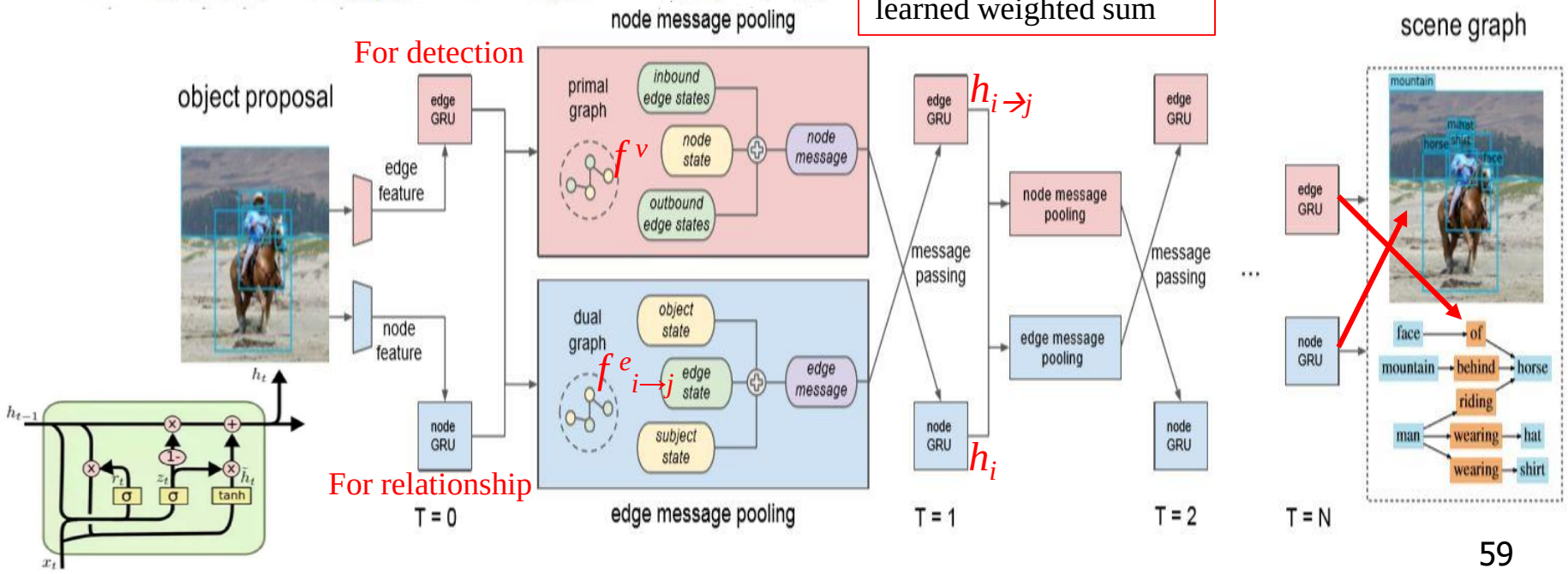


Step 2: IMP for Scene Graph

- The visual feature f^v (512-dim) of the 128 proposal box is used as feature for each node.
- The visual feature of the union box over the proposal boxes (b_i,b_j) as the visual feature
- $f^e_{i\to j}$ (512-dim) for edge $i \to j$. These visual features are extracted by an ROI pooling layer from the image.

$$\mathbf{x} = \{x_i^{cls}, x_i^{bbox}, x_{i\to j} | i = 1 \ldots n, j = 1 \ldots n, i \neq j\}$$

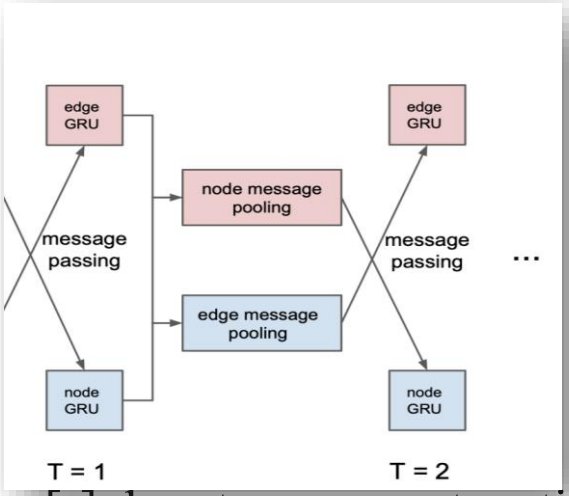
The $\oplus$ symbol denotes a learned weighted sum



Scene Graph Generation

- Each iteration, each GRU takes its previous hidden state and an incoming message as input to produce a new hidden state output.
- Each node and edge in the scene graph maintains its internal state in its corresponding GRU unit, where all nodes share the same GRU weights (node GRUs), and all edges share the other set of GRU weights (edge GRUs). This setup allows the model to pass messages (i.e., aggregation of GRU hidden states) among the GRU units along the scene graph topology.
- The model iteratively (N=40K) updates the hidden states of the GRUs. At the last iteration step, the hidden states of the GRUs are used to predict object categories, bounding box offsets, and relationship types.

Node/Edge GRUs Updates



From all possible edges

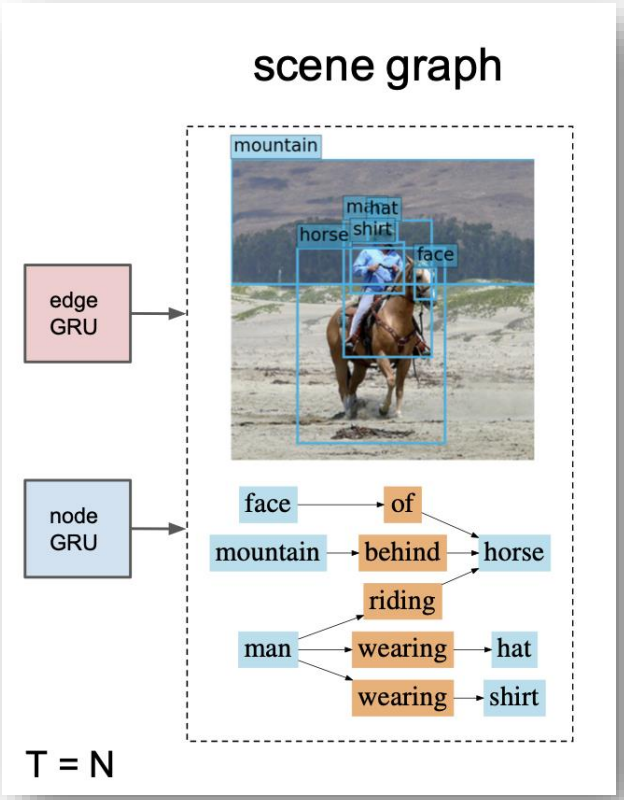
$$m_i = \sum_{j:i\to j} \sigma(\mathbf{v}_1^T[h_i, h_{i\to j}])h_{i\to j} + \sum_{j:j\to i} \sigma(\mathbf{v}_2^T[h_i, h_{j\to i}])h_{j\to i}$$

From only two nodes

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

- where $[\cdot]$ denotes a concatenation of vectors, and σ denotes a sigmoid function. $\mathbf{w_1}$, $\mathbf{w_2}$ and $\mathbf{v_1}$, $\mathbf{v_2}$ are learnable parameters
- During training, randomly select 128 boxes as the object proposals. For edges training, first sample all edges that have labels. If an image has less than 128 labeled edges, we fill the rest with unlabeled edges.
- At test time, we use NMS to select at most 50 boxes from the object proposals with an IoU threshold of 0.3. We make predictions on all edges except the self-connections at the test time.

Step 3: Predict Outputs

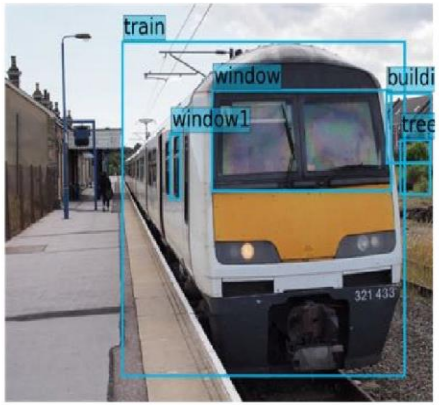


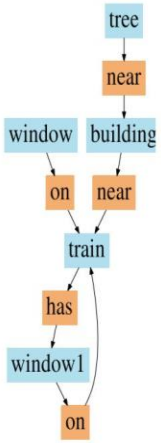
- Use hidden states to generate outputs
  - Object categories
    - FC + softmax, cross entropy loss
  - Bounding box offsets
    - FC, L1 loss
  - Relationship types
    - FC + softmax, cross entropy loss

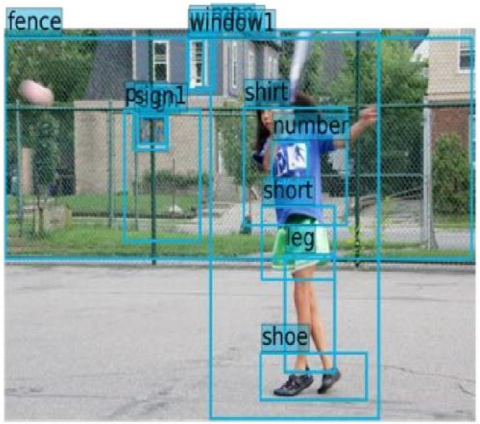
Scene Graph Dataset

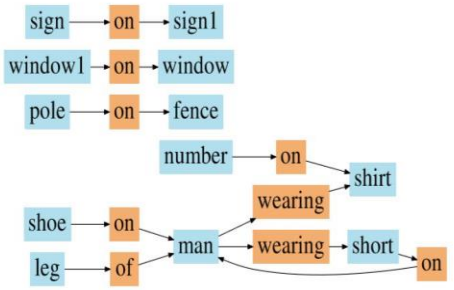
- Visual Genome dataset contains 108,077 images, an average of 25 distinct objects and 22 relationships per image.
- In this experiment, the most frequent 150 object categories and 50 predicates (pairwise relationship of a set of localized objects) for evaluation. As a result, each image has a scene graph of around 11.5 objects and 6.2 relationships. 70% of the images for training and the remaining 30% for testing.

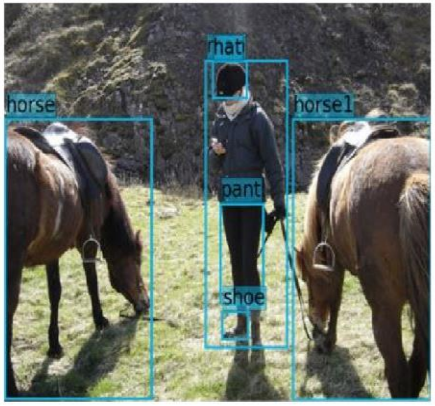
Predicate Classification

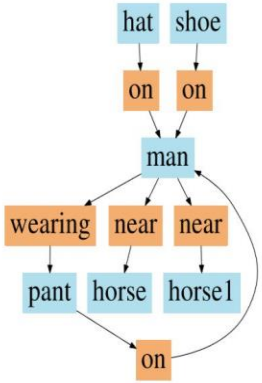


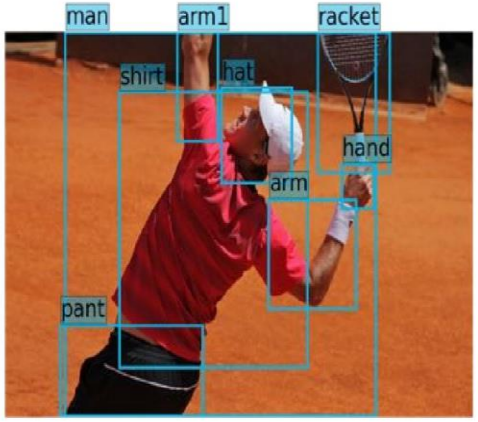


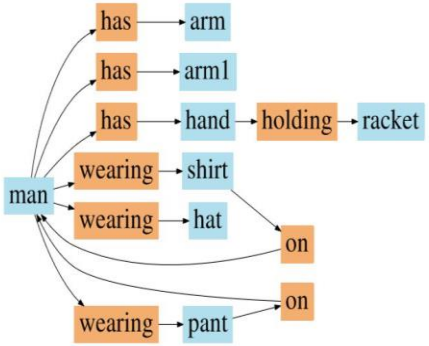












Predicate Classification

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

C. Lu, et al., "Visual relationship detection with language priors," ECCV 2016