上海科技大学
ShanghaiTech University

Project-1: Applications

Yujiao Shi
SIST, ShanghaiTech
Spring, 2024

Outline

上海科技大学
ShanghaiTech University

- Recurrent Neural Networks
  - Sequence modeling, Autoregressive models
  - (Vanilla) RNN models
  - LSTM
- Application : Optical Flow Estimation
  - FlowNet
  - Spynet
  - PWC-Net
  - RAFT

Sequence modeling

上海科技大学 ShanghaiTech University

- Modeling a sequence of tokens
  - Running example: sentences
- Goal: learn/build a good distribution of sentences
- Inputs: a corpus of sentences $\mathbf{s}^{(1)}, \cdots, \mathbf{s}^{(N)}$
- Output: a distribution $p(\mathbf{s})$
- Common approach: maximum likelihood
  - Assume sentences are independent

$$\max \prod_{i=1}^{N} p(\mathbf{s}^{(i)})$$

Sequence modeling

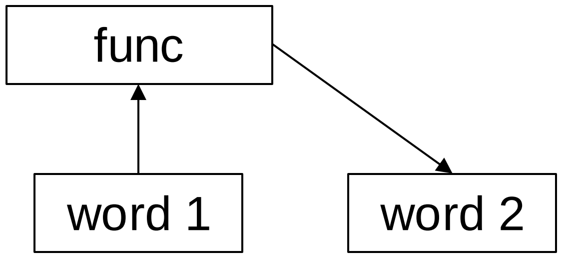
- What is $p(\mathbf{s})$?
- A sentence is a sequence of words $w_1, w_2, \cdots, w_T$.

$$p(\mathbf{s}) = p(w_1, \ldots, w_T) = p(w_1)p(w_2 \mid w_1) \cdots p(w_T \mid w_1, \ldots, w_{T-1}).$$

  - Essentially aim to predict the next word
- Markovian assumption
  - The distribution over the next word depends on the preceding few words. For example,

$$p(w_t \mid w_1, \ldots, w_{t-1}) = p(w_t \mid w_{t-3}, w_{t-2}, w_{t-1}).$$

  - Autoregressive model
    - Memoryless
    - Can be modeled by a parametrized function



Traditional language models

- N-Gram model
  - Autoregressive model: Markov assumption
  - Use a conditional probability table

	cat	and	city	$\cdots$
the fat	0.21	0.003	0.01	
four score	0.0001	0.55	0.0001	$\cdots$
New York	0.002	0.0001	0.48	
$\vdots$		$\vdots$		

  - Estimate the probabilities from the empirical distribution

$$p(w_3 = \mathrm{cat} \mid w_1 = \mathrm{the}, w_2 = \mathrm{fat}) = \frac{\mathrm{count}(\mathrm{the\ fat\ cat})}{\mathrm{count}(\mathrm{the\ fat})}$$

  - The phrases we're counting are called n-grams (where n is the length), so this is an n-gram language model.
    - Note: the above example is considered a 3-gram model, not a 2-gram model!

Traditional language models

- Problems with n-gram language models
  - The number of entries in the conditional probability table is exponential in the context length
  - Data sparsity: most n-grams never appear in the corpus
- Solutions
  - Use a short context (less expressive)
  - Smooth the probabilities (priors)
  - Using an ensemble of n-gram models with different n

Neural language model

上海科技大学 ShanghaiTech University

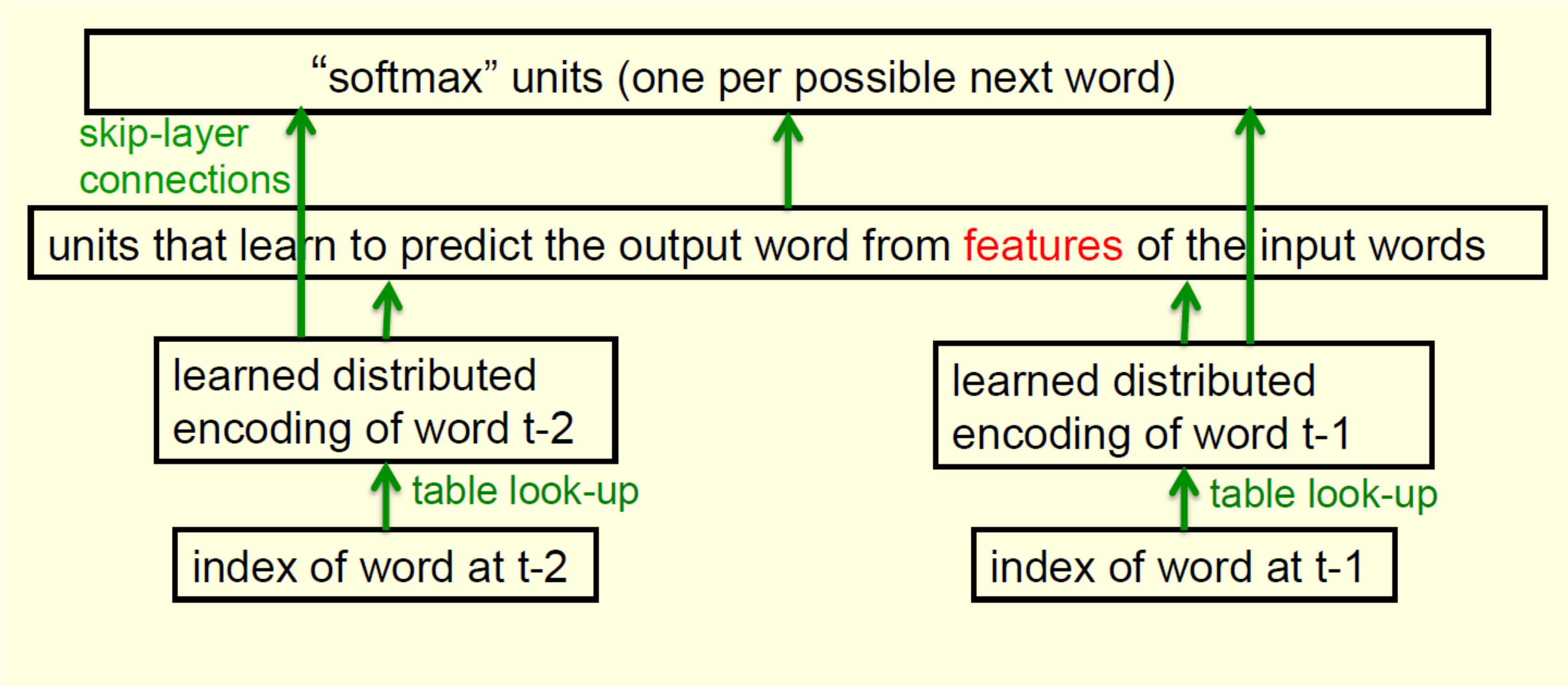
- Predicting the distribution of the next word given the previous K is a multiway classification problem
  - Inputs: previous K words
  - Output/Target: next word
  - Loss: cross-entropy

$$
\begin{aligned}
-\log p(\mathbf{s}) &= -\log \prod_{t=1}^{T} p(w_t \mid w_1, \ldots, w_{t-1}) \\
&= -\sum_{t=1}^{T} \log p(w_t \mid w_1, \ldots, w_{t-1}) \\
&= -\sum_{t=1}^{T} \sum_{v=1}^{V} t_{tv} \log y_{tv},
\end{aligned}
$$

where t_{iv} is the one-hot encoding for the ith word and y_{iv} is the predicted probability for the ith word being index v.

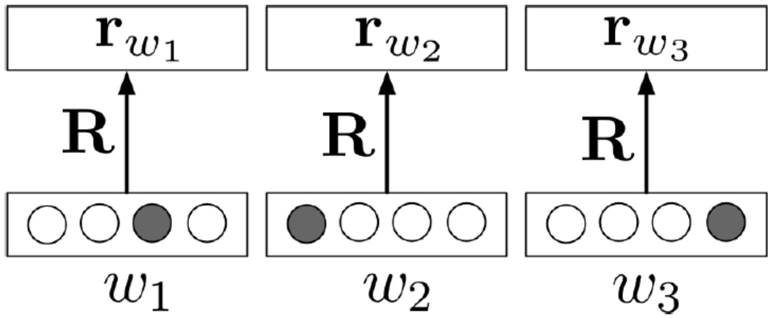
Neural language model

- Model structure (context length = 2)



Neural language model

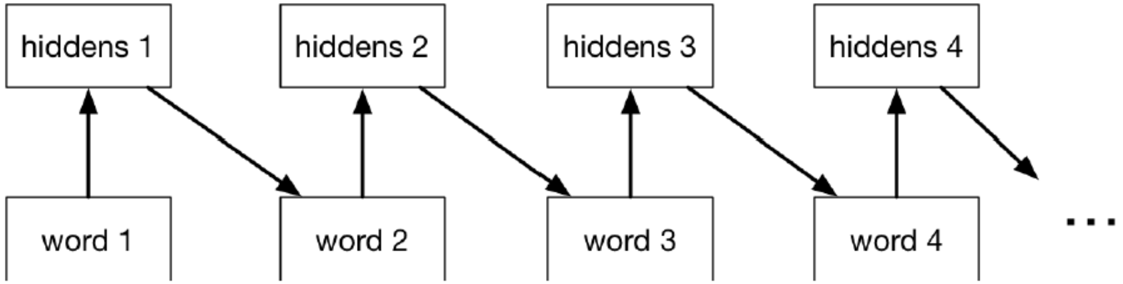
- Word embedding
  - If we use a 1-of-K encoding for the words, the first layer can be thought of as a linear layer with tied weights.



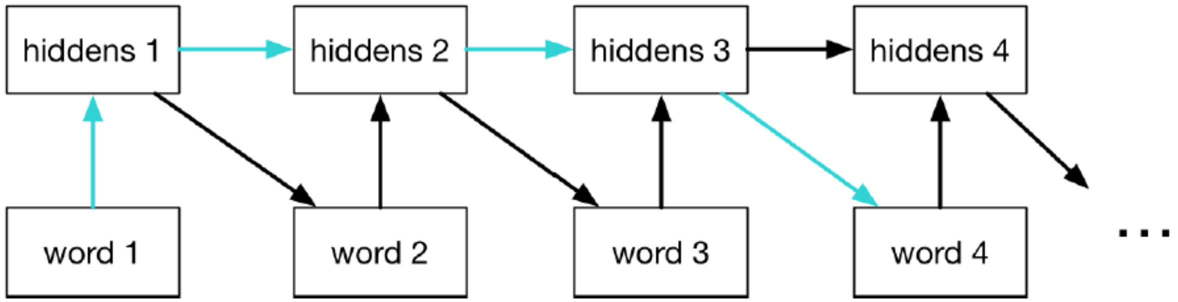
  - The weight matrix basically acts like a lookup table. Each column is the representation of a word, also called an embedding, feature vector, or encoding.
    - "Embedding" emphasizes that it's a location in a high-dimensonal space; words that are closer together are more semantically similar
    - "Feature vector" emphasizes that it's a vector that can be used for making predictions, just like other feature mappigns we've looked at (e.g. polynomials)

Sequence modeling

- Problems?
- Autoregressive models are memoryless
  - Can only use information from their immediate context

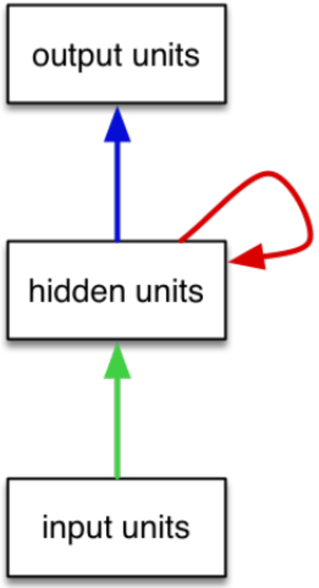


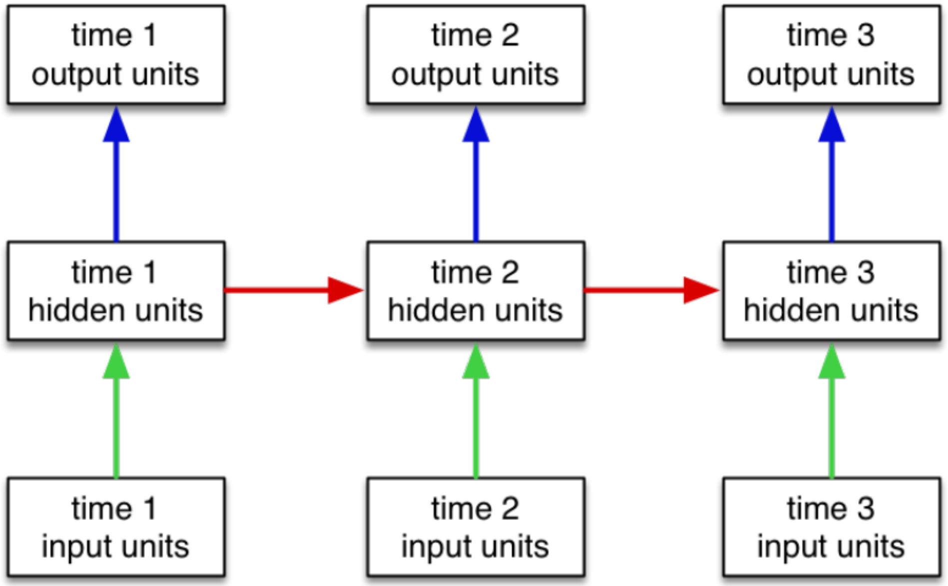
- Adding connections between hidden units
  - Having a memory lets the model use longer-term dependencies



Recurrent Neural Network

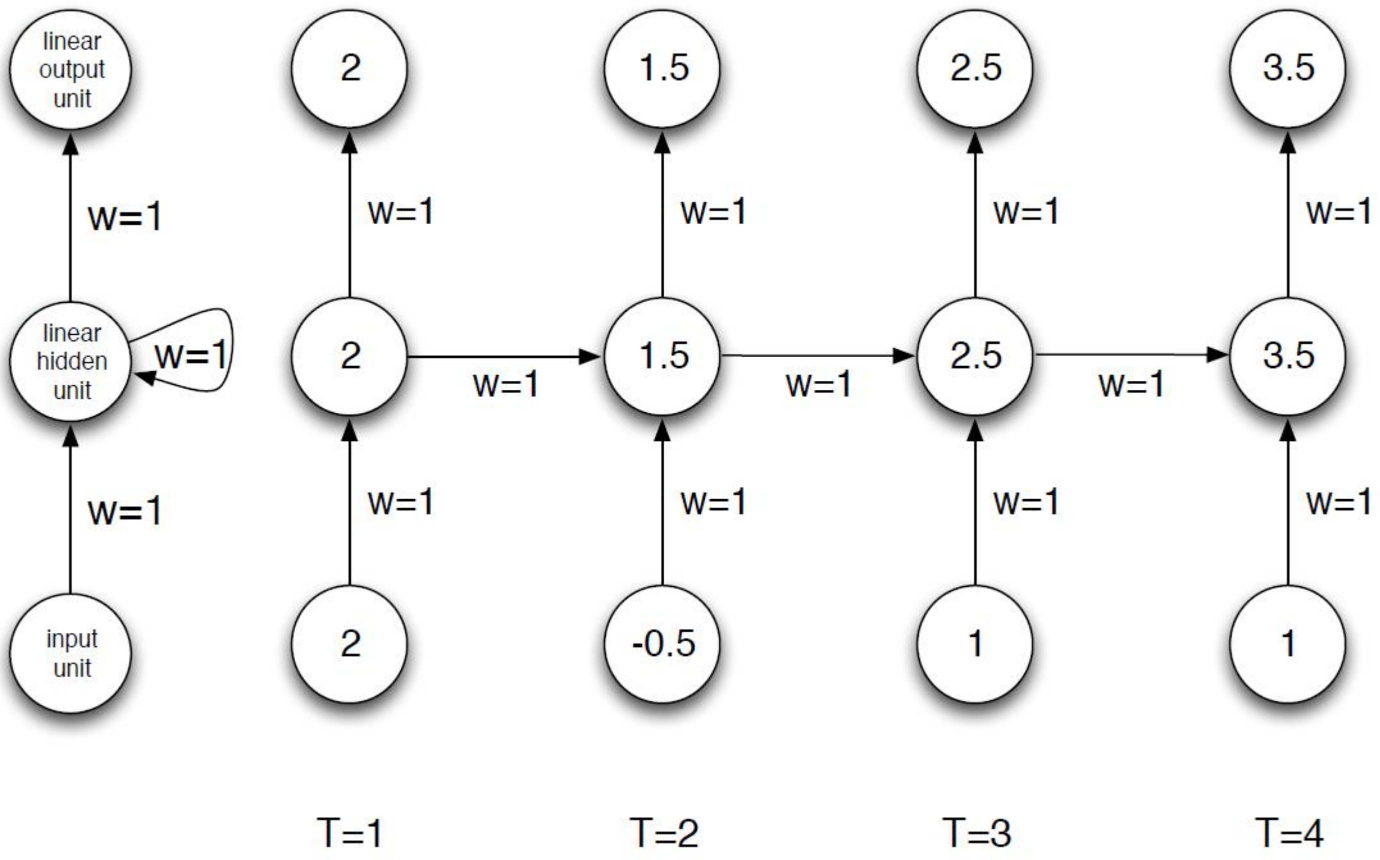
- Recurrent Neural Network as a dynamical system with one set of hidden units feeding into themselves
  - The network's graph has self-loops
- The RNN's graph can be unrolled by explicitly representing the units at all time steps
  - The weights and biases are shared





RNN examples

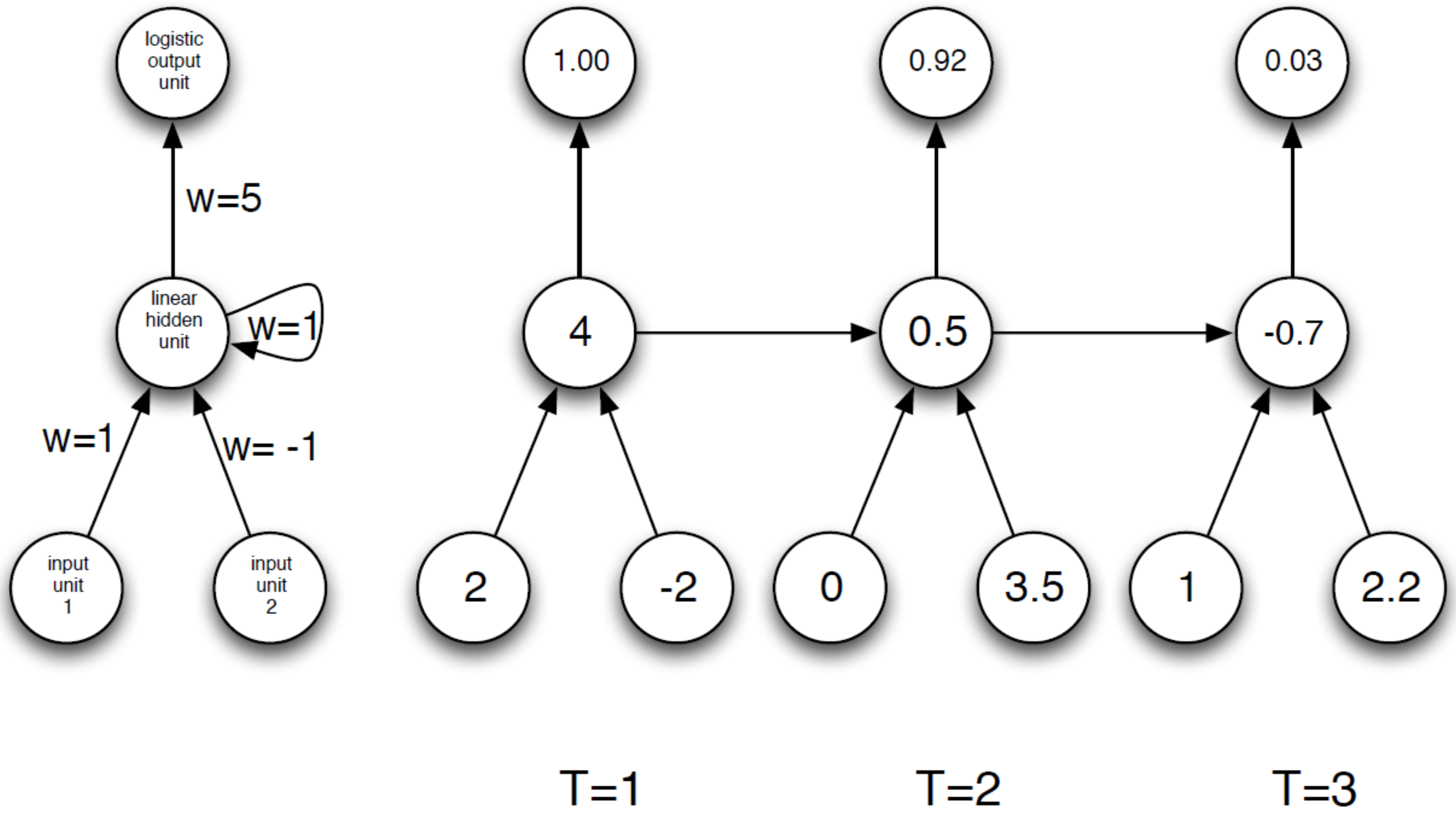
- Summation network



RNN examples

上海科技大学
ShanghaiTech University

- Summation & comparison network



Recurrent Neural Network

- General formulation

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

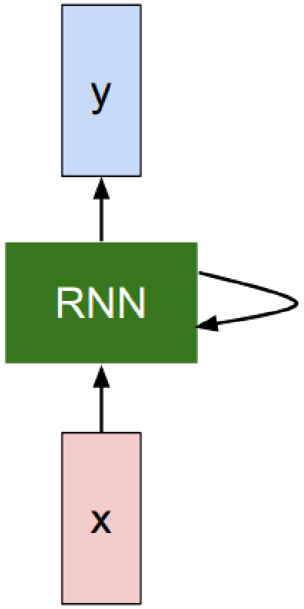
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at some time step

some function with parameters W



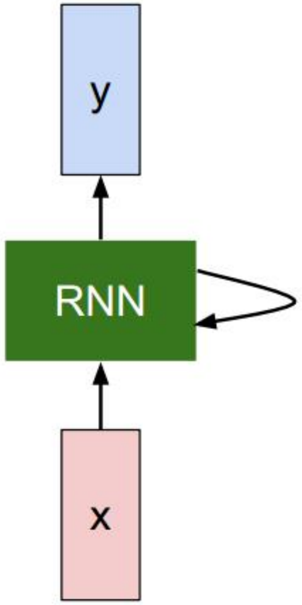
Recurrent Neural Network

- General formulation

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

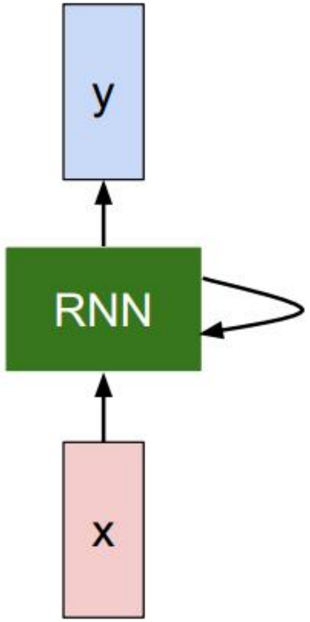
Notice: the same function and the same set of parameters are used at every time step.



(Vanilla)Recurrent Neural Network

- General formulation

The state consists of a single *"hidden"* vector **h**:



$$h_t = f_W(h_{t-1}, x_t)$$

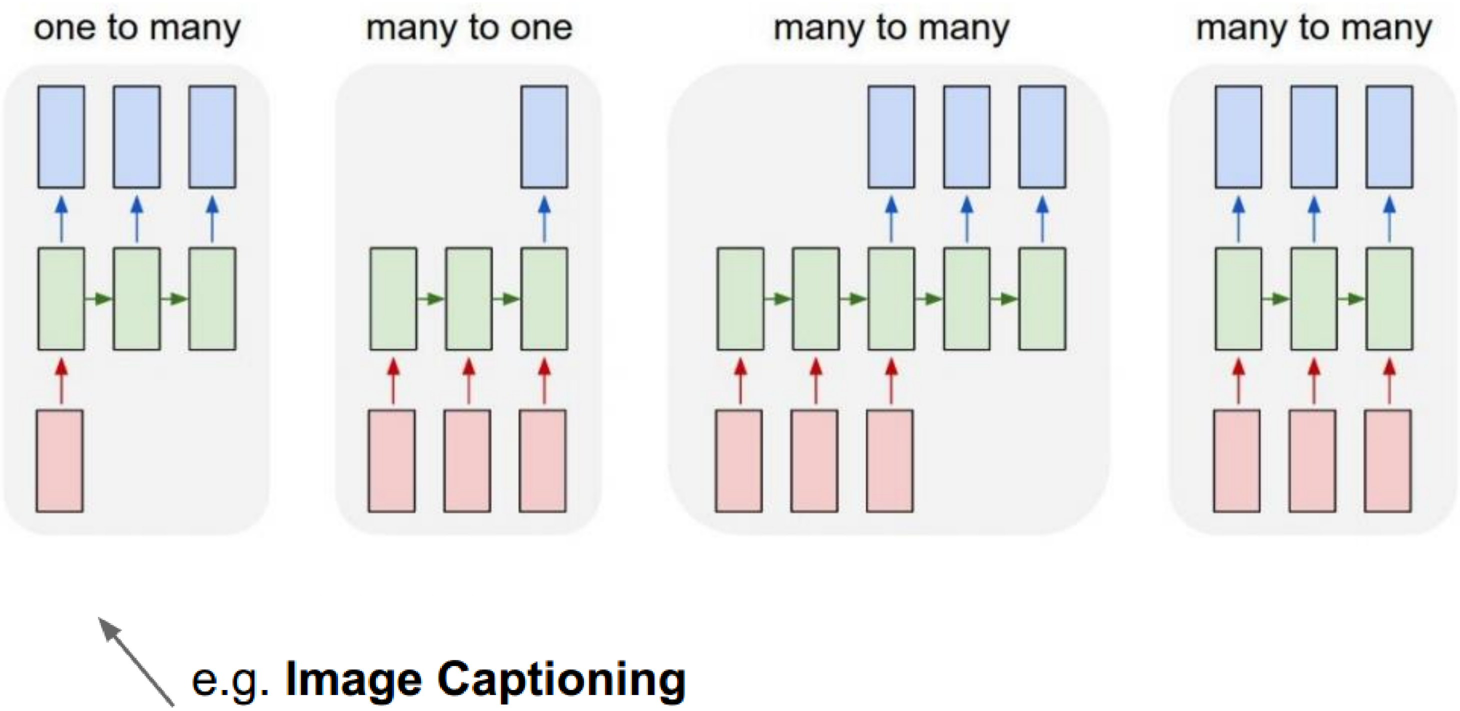
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

Recurrent Neural Network

上海科技大学 ShanghaiTech University

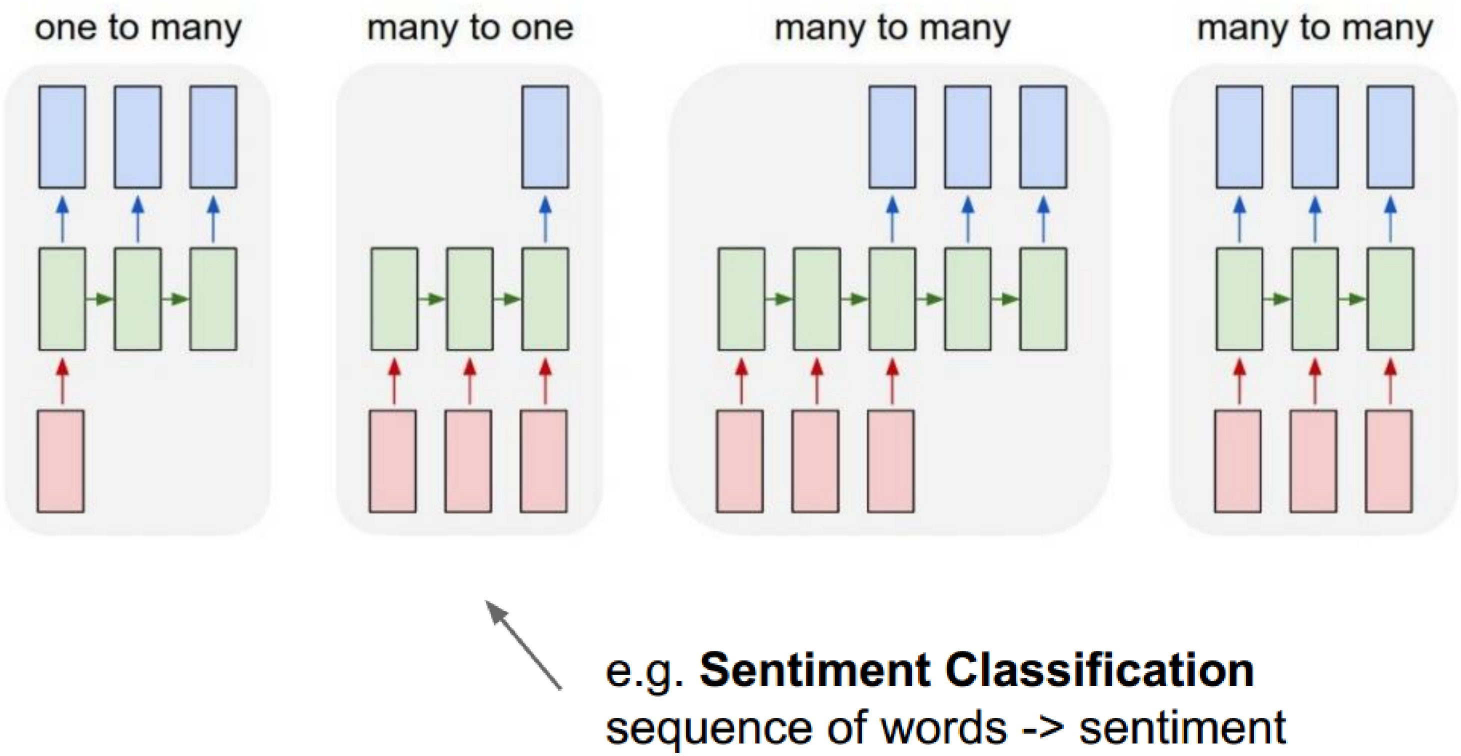
- Recurrent Neural Networks: model variants



e.g. **Image Captioning**
image -> sequence of words

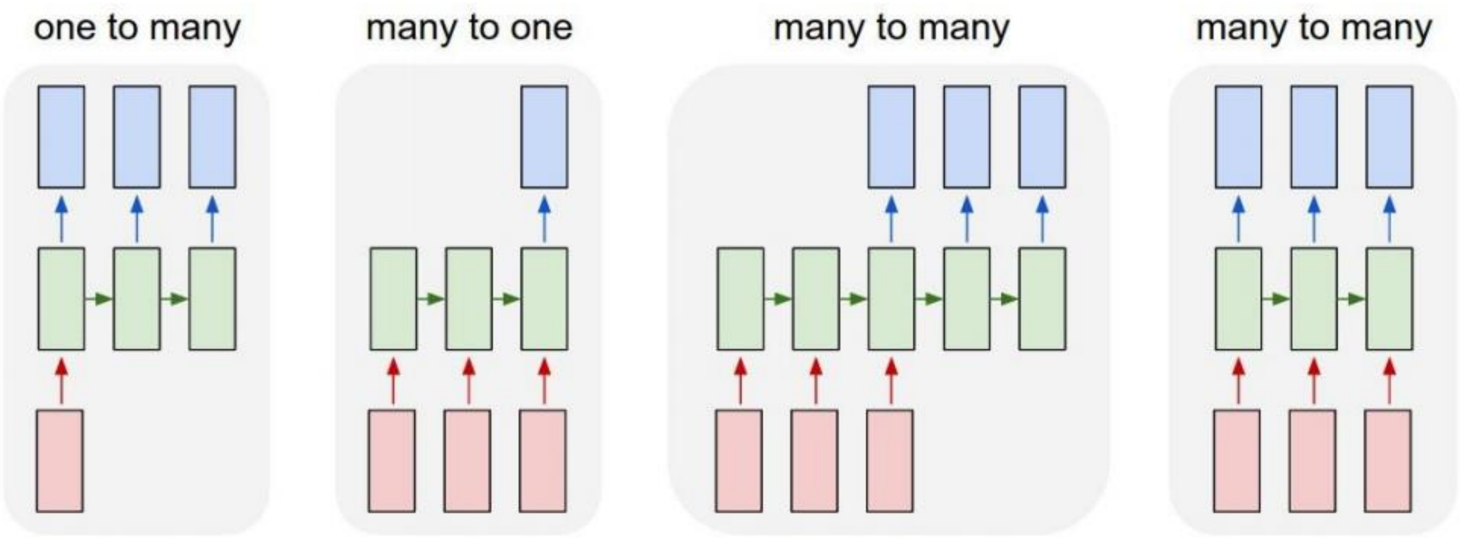
Recurrent Neural Network

- Recurrent Neural Networks: model variants



Recurrent Neural Network

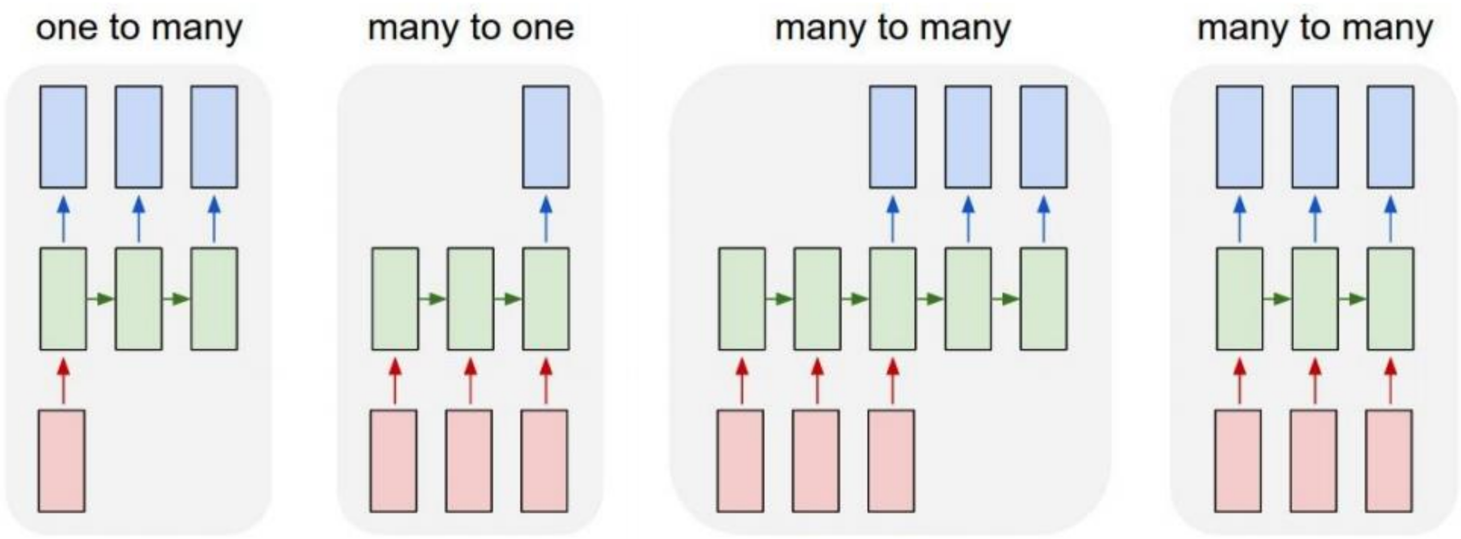
- Recurrent Neural Networks: model variants



e.g. **Machine Translation**
seq of words -> seq of words

Recurrent Neural Network

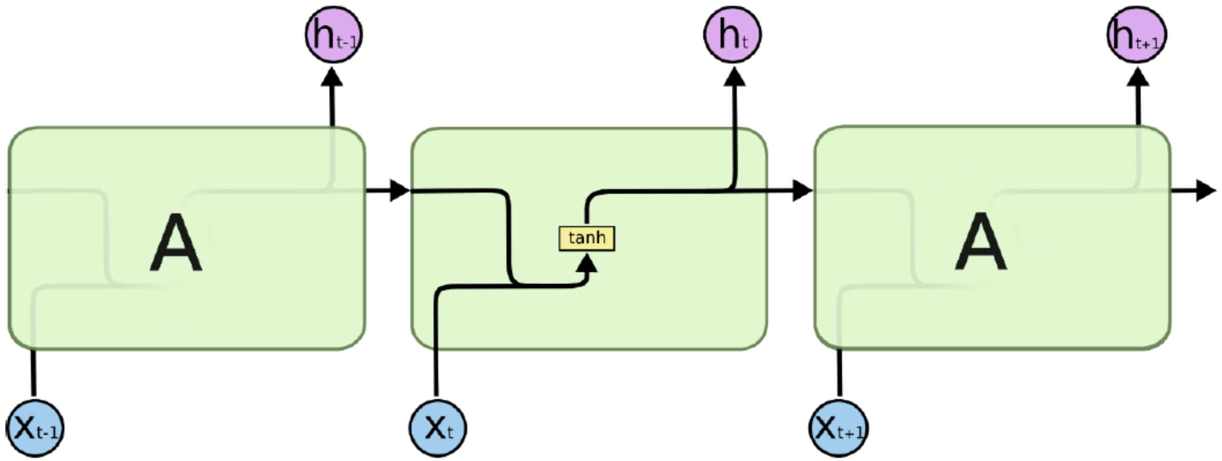
- Recurrent Neural Networks: model variants



e.g. **Video classification on frame level**

Standard RNN

- Recall

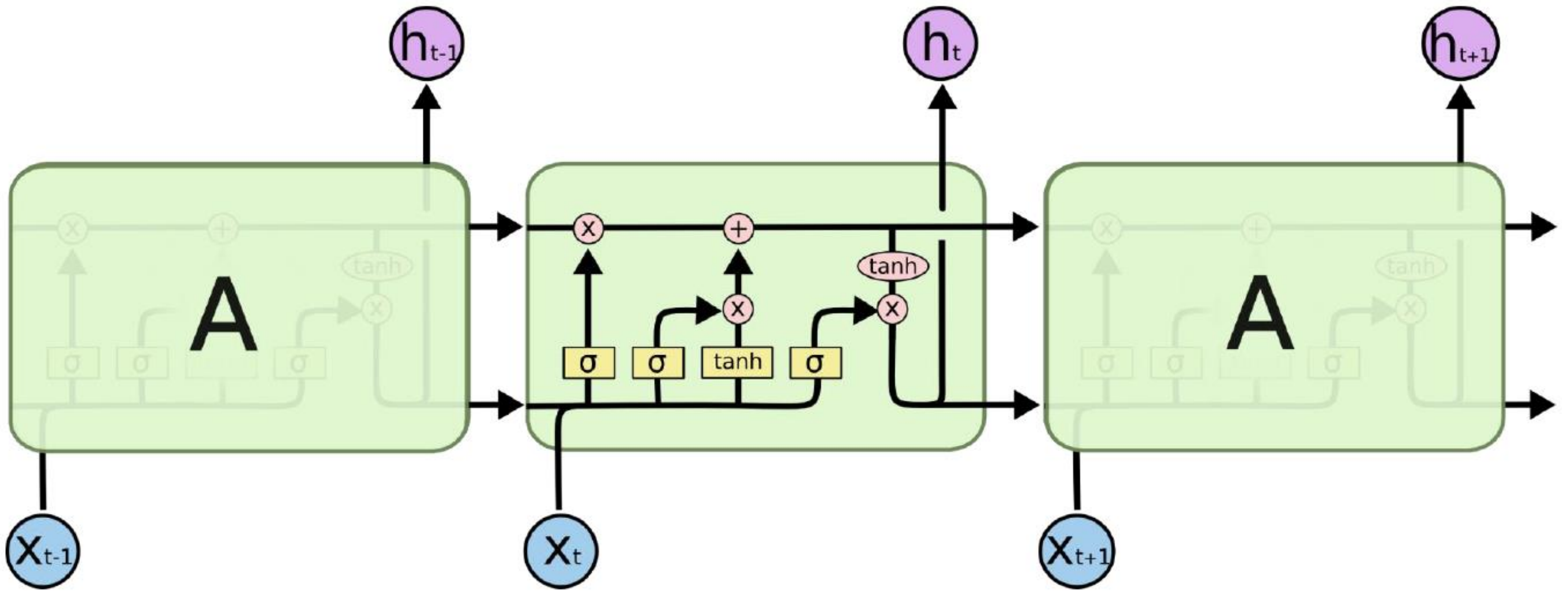


- Each recurrent neuron receives past outputs and current input
- Pass through a tanh function

Long Short Term Memory(LSTM)

上海科技大学
ShanghaiTech University

- LSTM uses multiplicative gates that decide if something is important or not

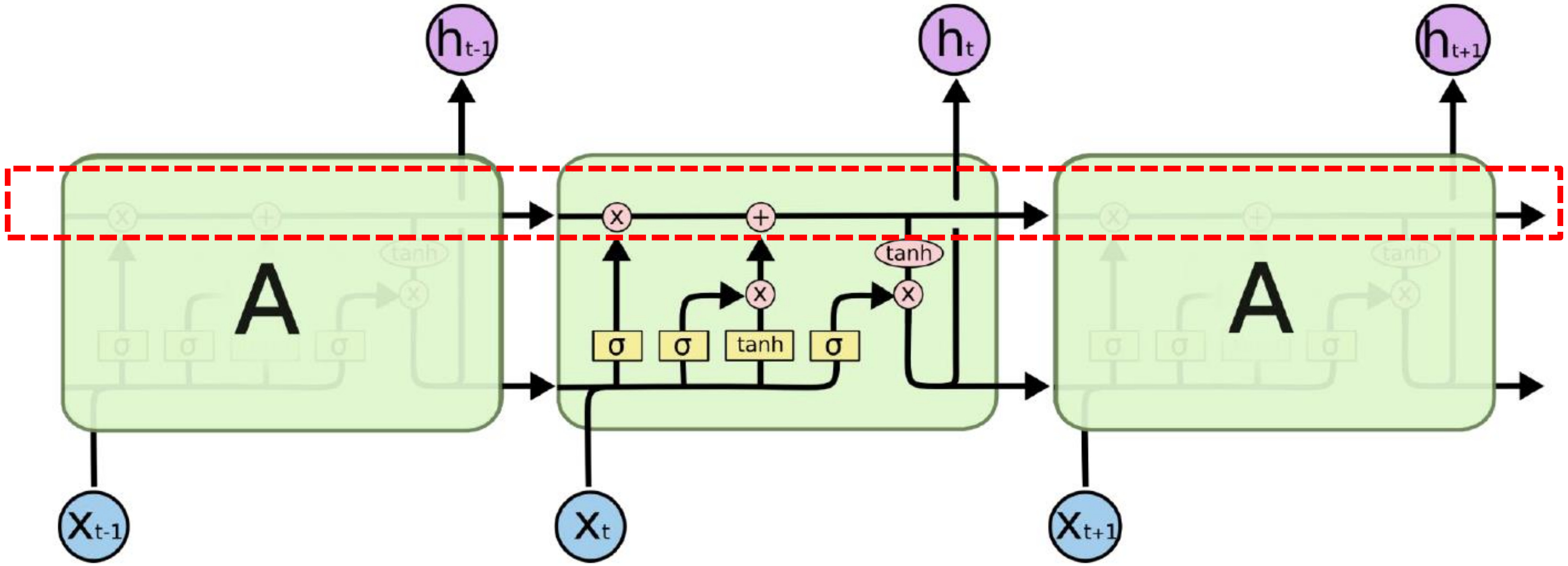


Hochreiter and Schmidhuber, “Long Short Term Memory”, Neural Computation

Long Short Term Memory(LSTM)

上海科技大学 ShanghaiTech University

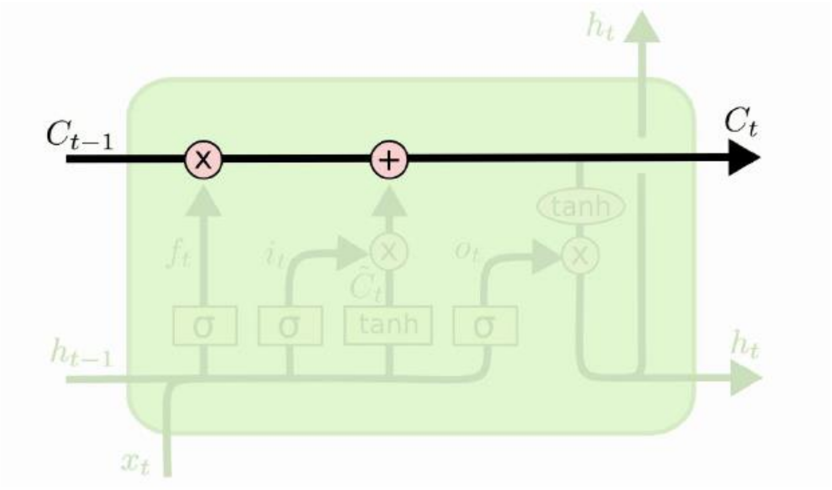
- Key component: a remembered cell state



Hochreiter and Schmidhuber, “Long Short Term Memory”, Neural Computation

LSTM: cell state

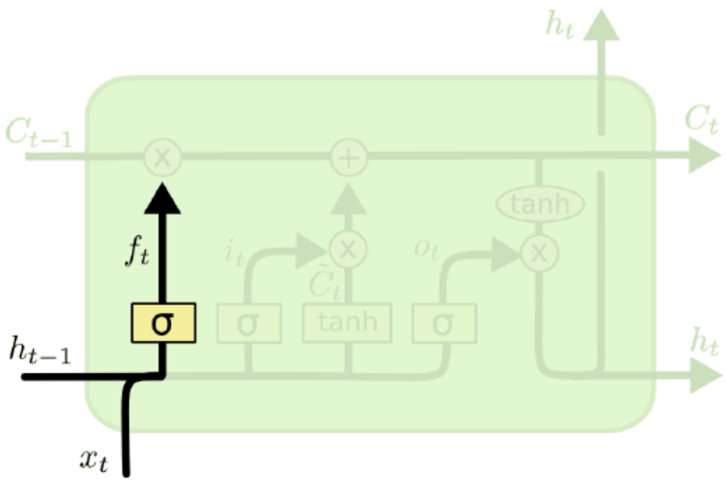
- A linear history
  - Carries information through
  - Only affected by a gate and addition of current information, which is also gated



LTSM: gates

LSTM: forget gate

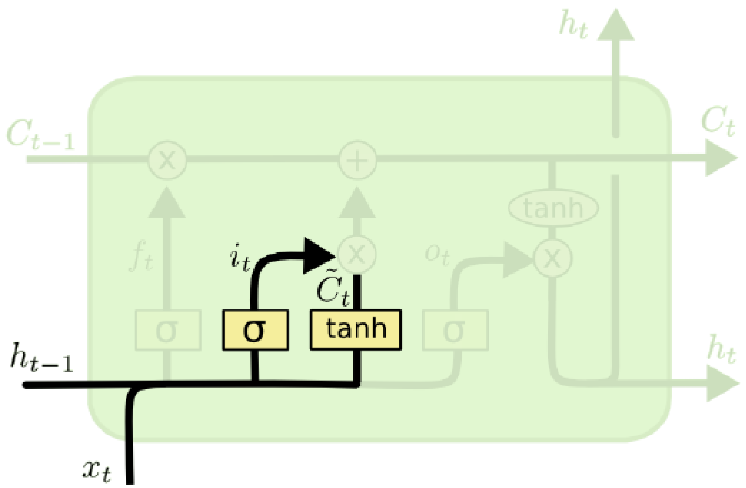
- The first gate determines whether to carry over the history or to forget it
  - Soft decision: how much of the history C_{t-1} to carry over
  - Determined by the current input x_t and the previous state h_{t-1}
  - $\langle h_{t-1}, C_{t-1} \rangle$ can be viewed as partial key-value pairs



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

LSTM: input gate

- The second gate has two parts
  - A gate that decides if it is worth remembering
  - A nonlinear transformation that extracts new and interesting information from the input
  - Both use the current input and the previous state

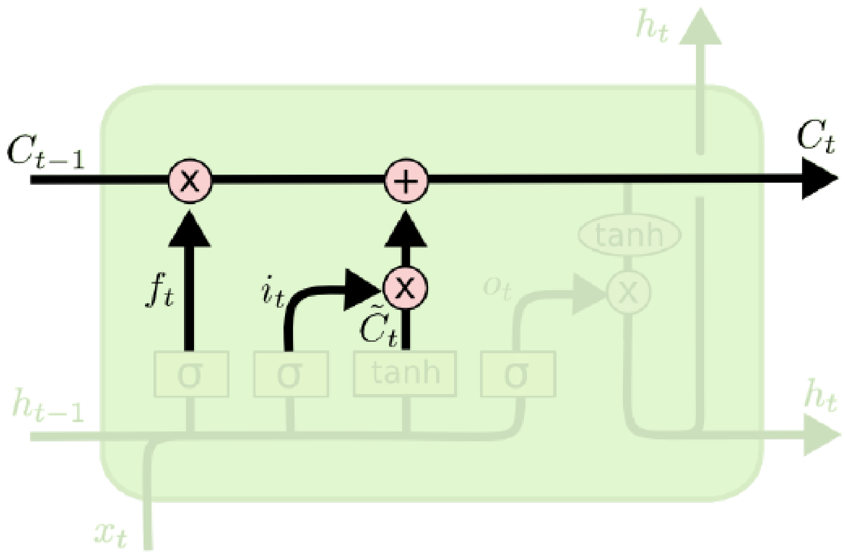


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

LSTM: Memory cell update

- The output of the second part is added into the current memory cell
  - Can be viewed as value update in a key-value pair
  - The input and state jointly decide how much of history info is kept and how much of embedded input info is added
  - A dynamic mixture of experts at each time step

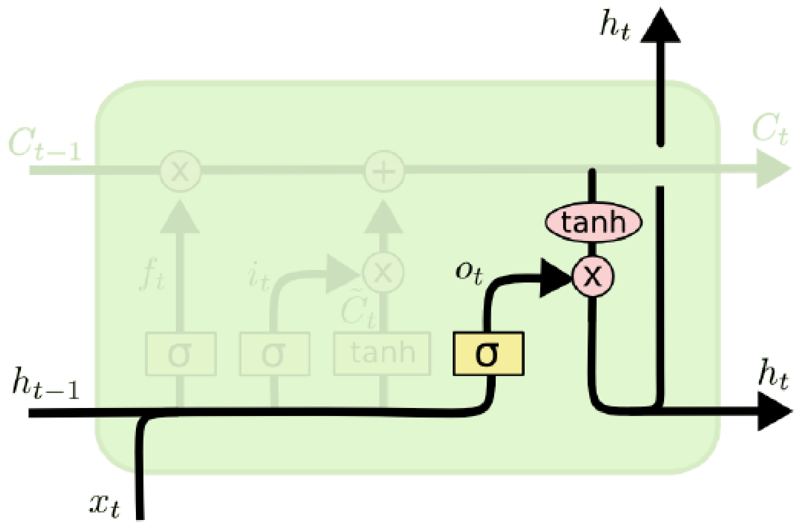


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM: Output gate

上海科技大学
ShanghaiTech University

- The third gate is the output gate
  - To decide if the memory cell contents are worth reporting at this time using the current input and previous state
- The output of the cell or the state
  - A nonlinear transform of the cell values
  - Compress it with tanh to make it in (-1,1)
  - Note the separation of key-value representation



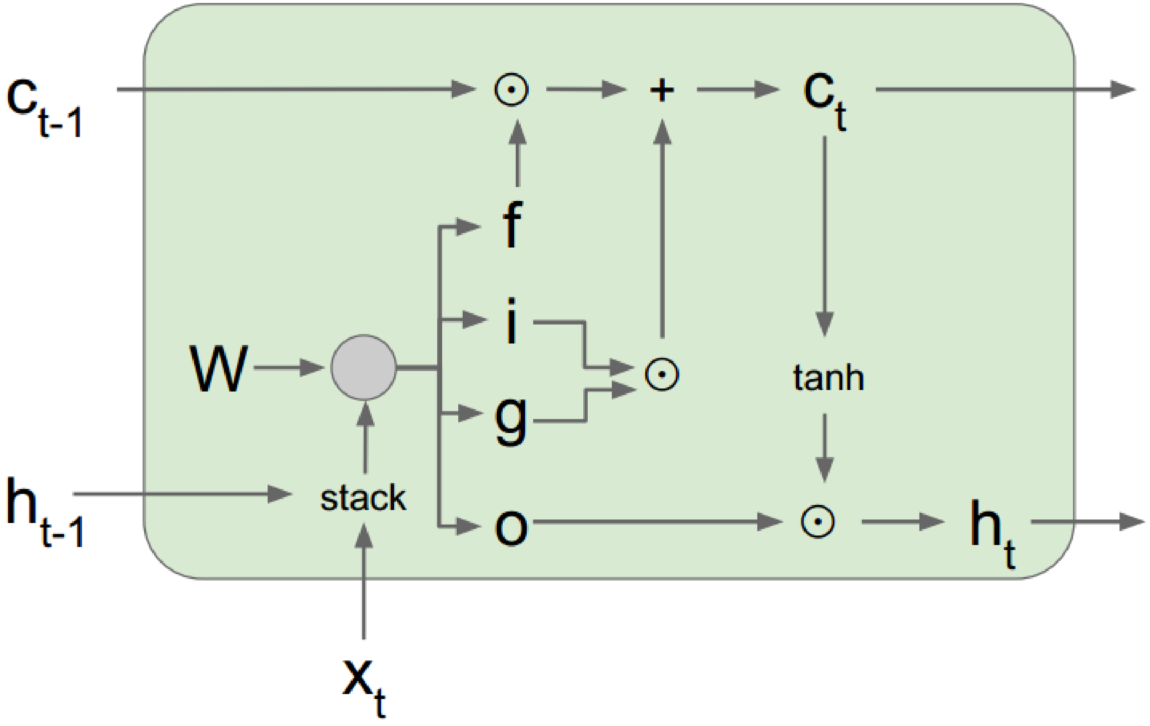
$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Long Short Term Memory(LSTM)

上海科技大学
ShanghaiTech University

[Hochreiter et al., 1997]



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

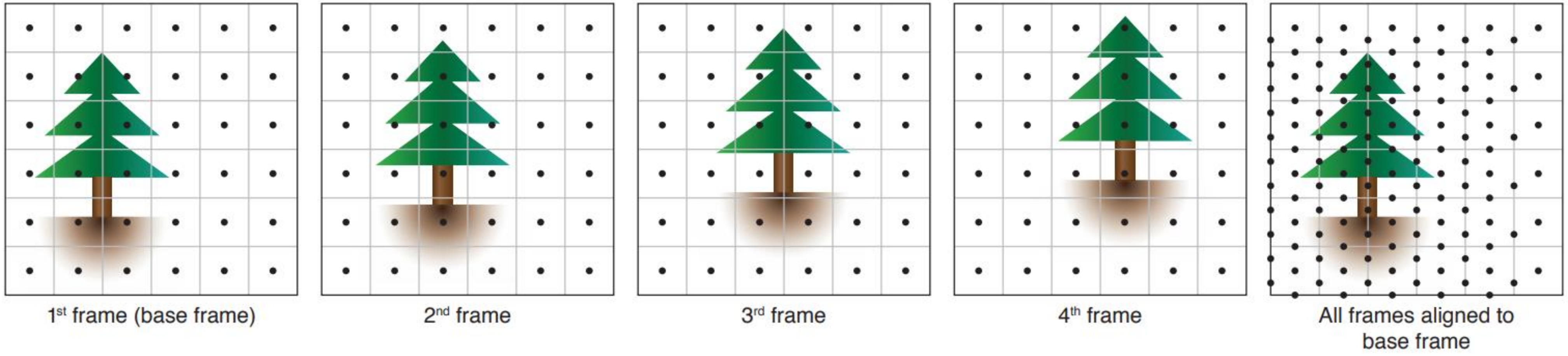
$$h_t = o \odot \tanh(c_t)$$

Outline

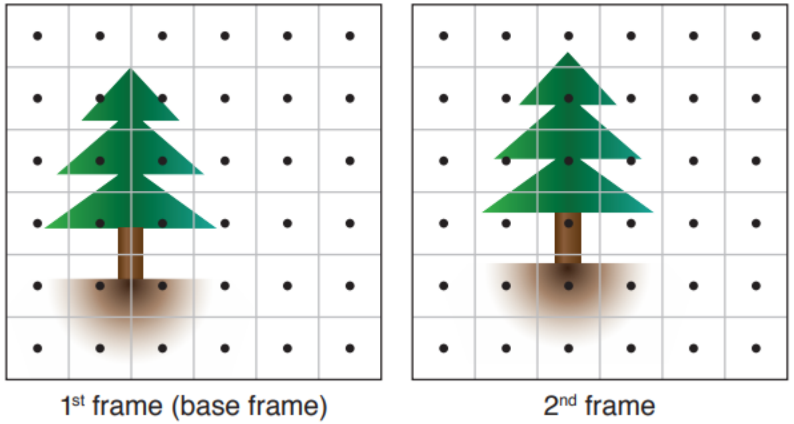
- Recurrent Neural Networks
  - Sequence modeling, Autoregressive models
  - (Vanilla) RNN models
  - LSTM
- Application : Optical Flow Estimation
  - FlowNet
  - Spynet
  - PWC-Net
  - RAFT

What is Optical Flow?

上海科技大学 ShanghaiTech University



How to estimate Optical Flow (OF) between two images?

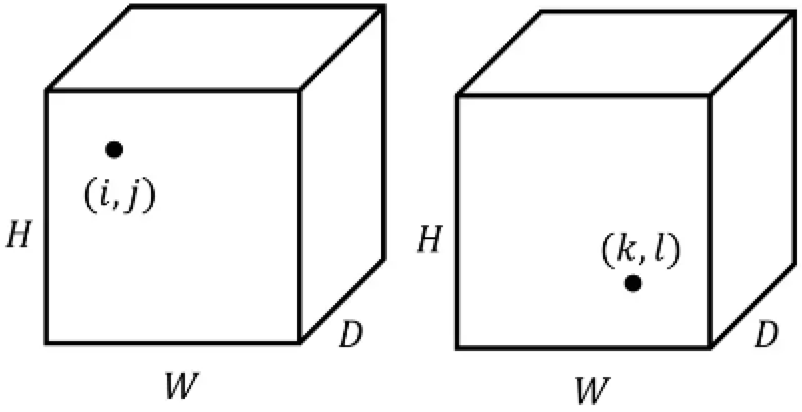


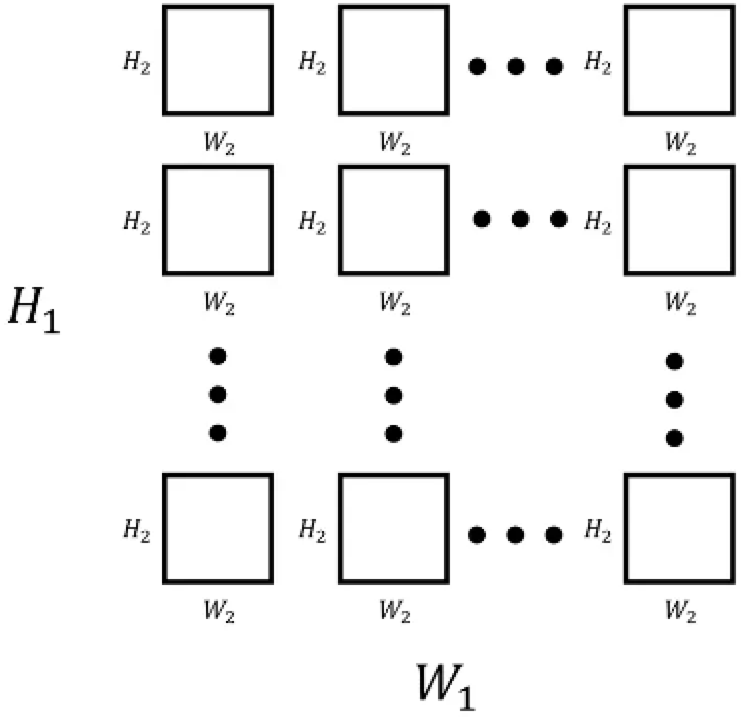
OF Estimation Basics -- Correlation

上海科技大学 ShanghaiTech University

Cost Volume: $\mathbf{C}(g_\theta(I_1), g_\theta(I_2)) \in \mathbb{R}^{H \times W \times H \times W}$

$$C_{ijkl} = \sum_h g_\theta(I_1)_{ijh} \cdot g_\theta(I_2)_{klh}$$





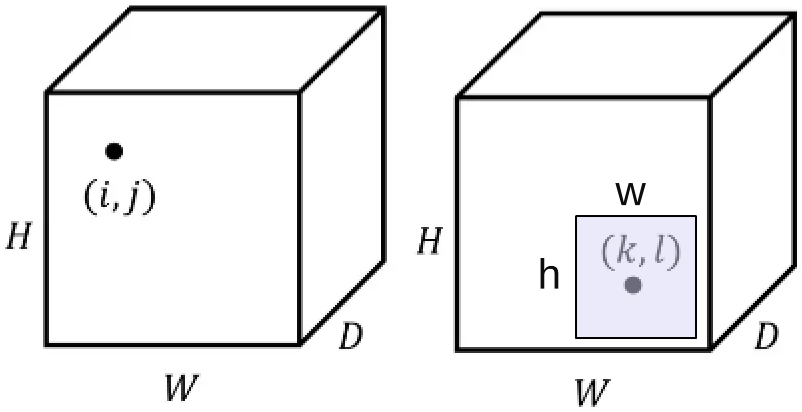
Global Cost Volume (with shape of H x W x H x W):

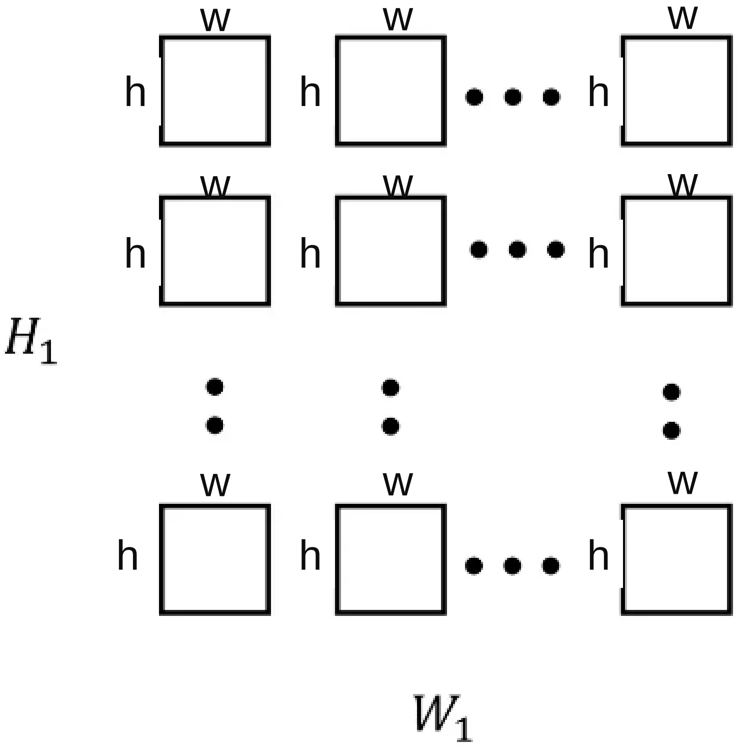
Disaster for memory and computation if H and W are large.

OF Estimation Basics -- Correlation

Cost Volume: $\mathbf{C}(g_\theta(I_1), g_\theta(I_2)) \in \mathbb{R}^{H \times W \times h \times w}$

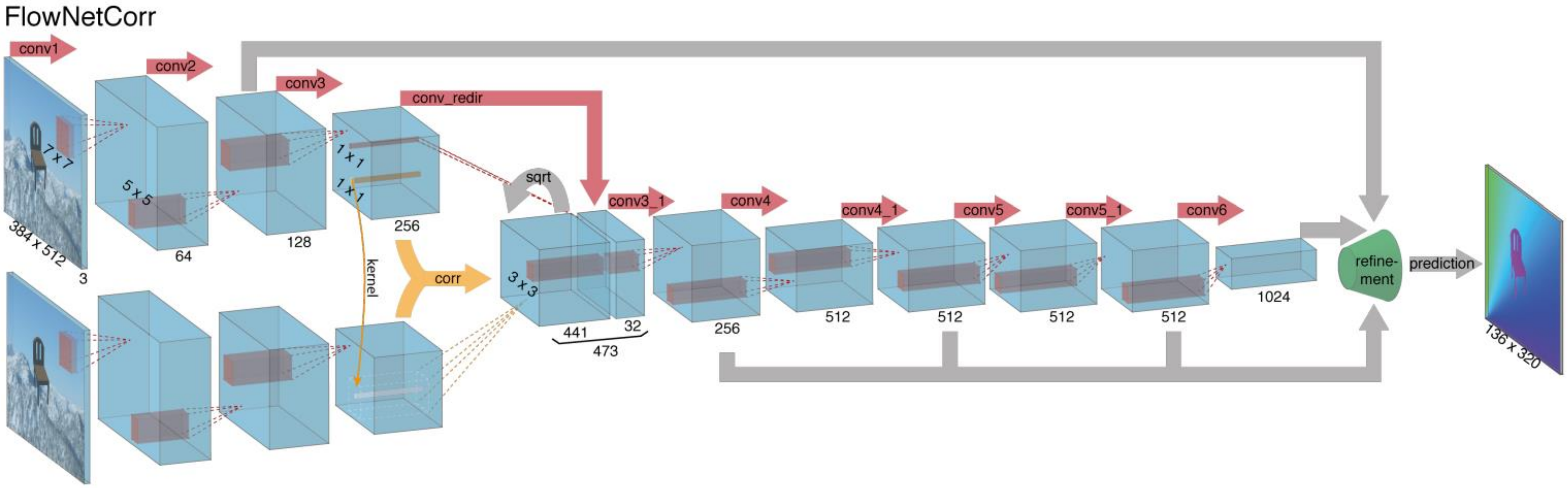
$$C_{ijkl} = \sum_h g_\theta(I_1)_{ijh} \cdot g_\theta(I_2)_{klh}$$





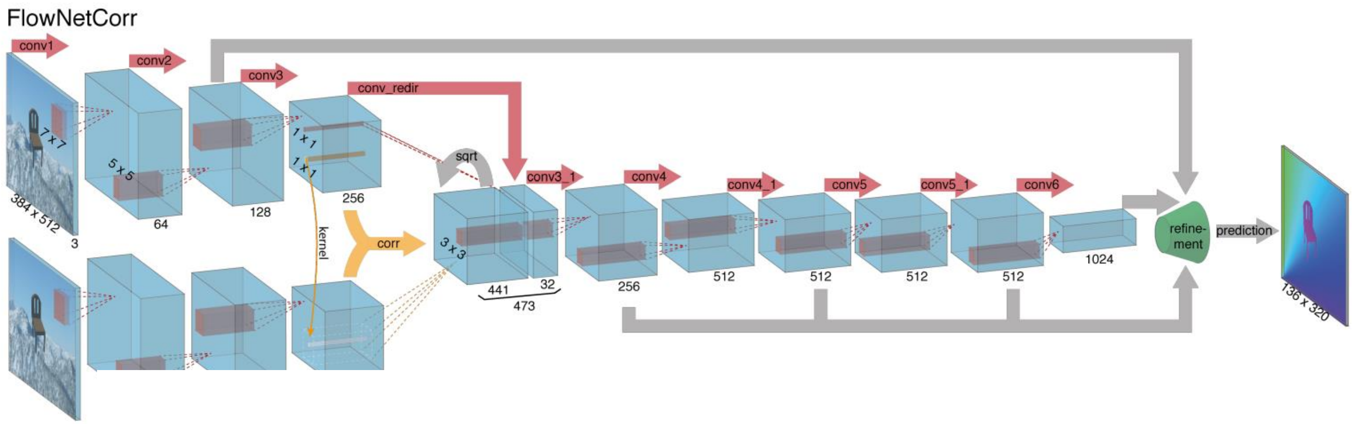
Local Cost Volume (with shape of H x W x h x w):

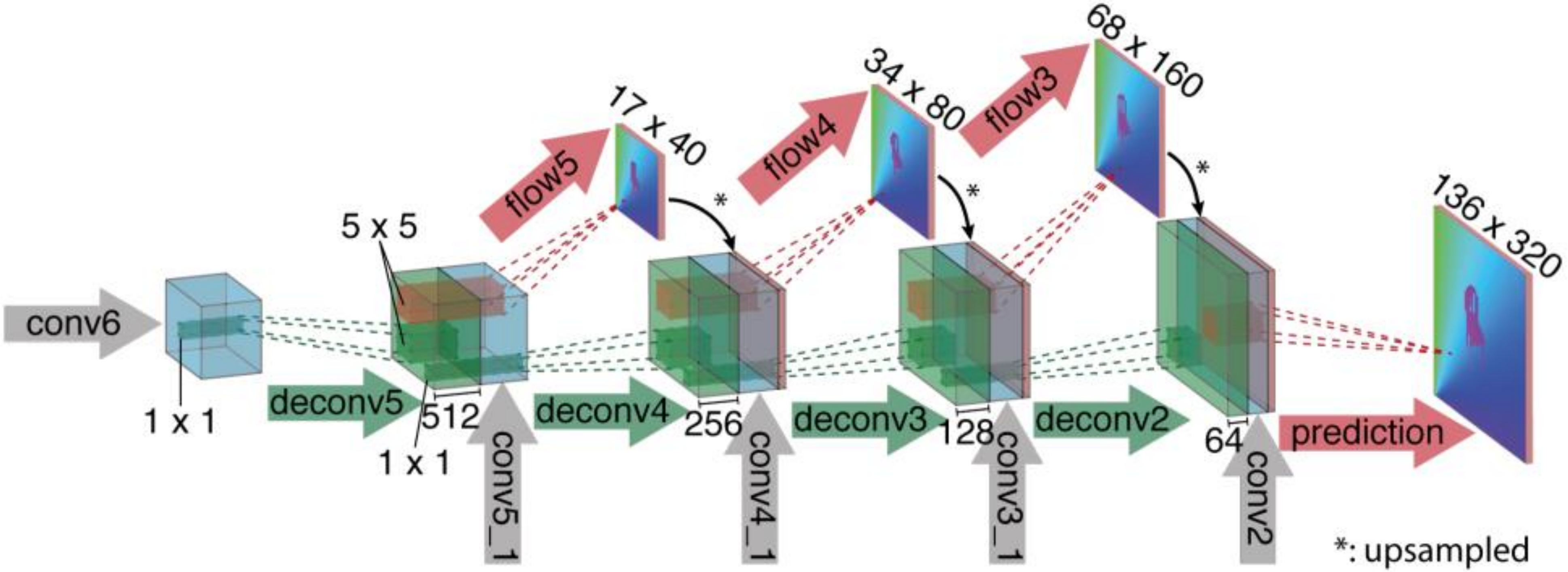
FlowNet (ICCV 2015)



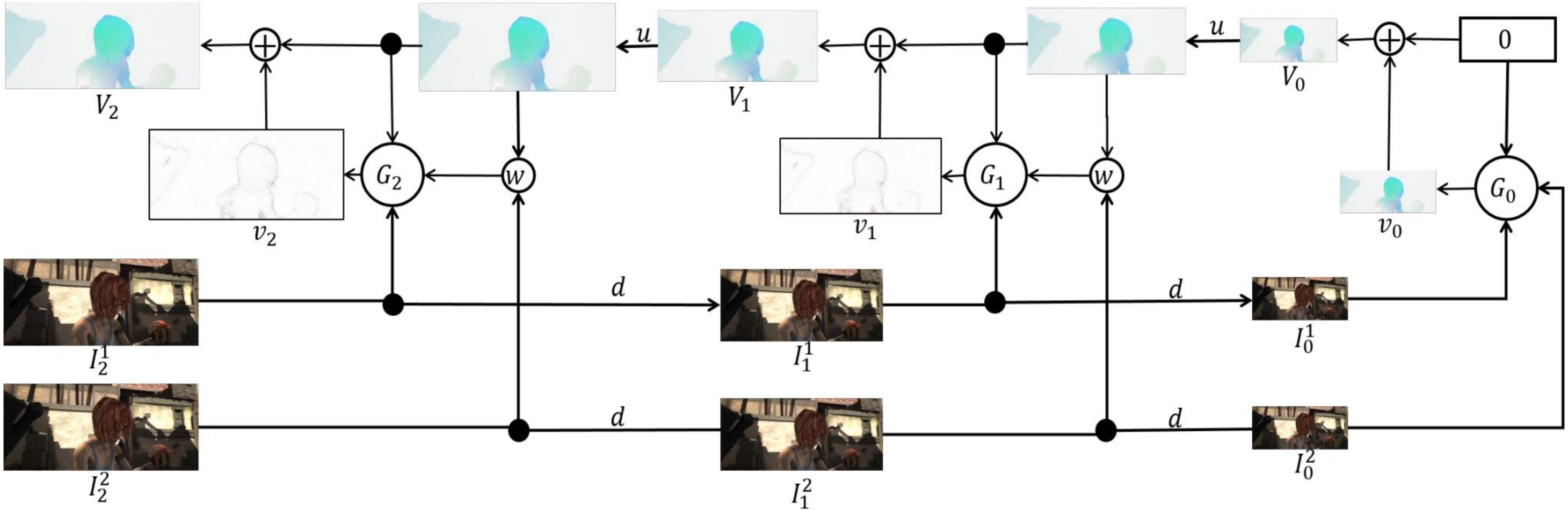
FlowNet (ICCV 2015)

上海科技大学
ShanghaiTech University

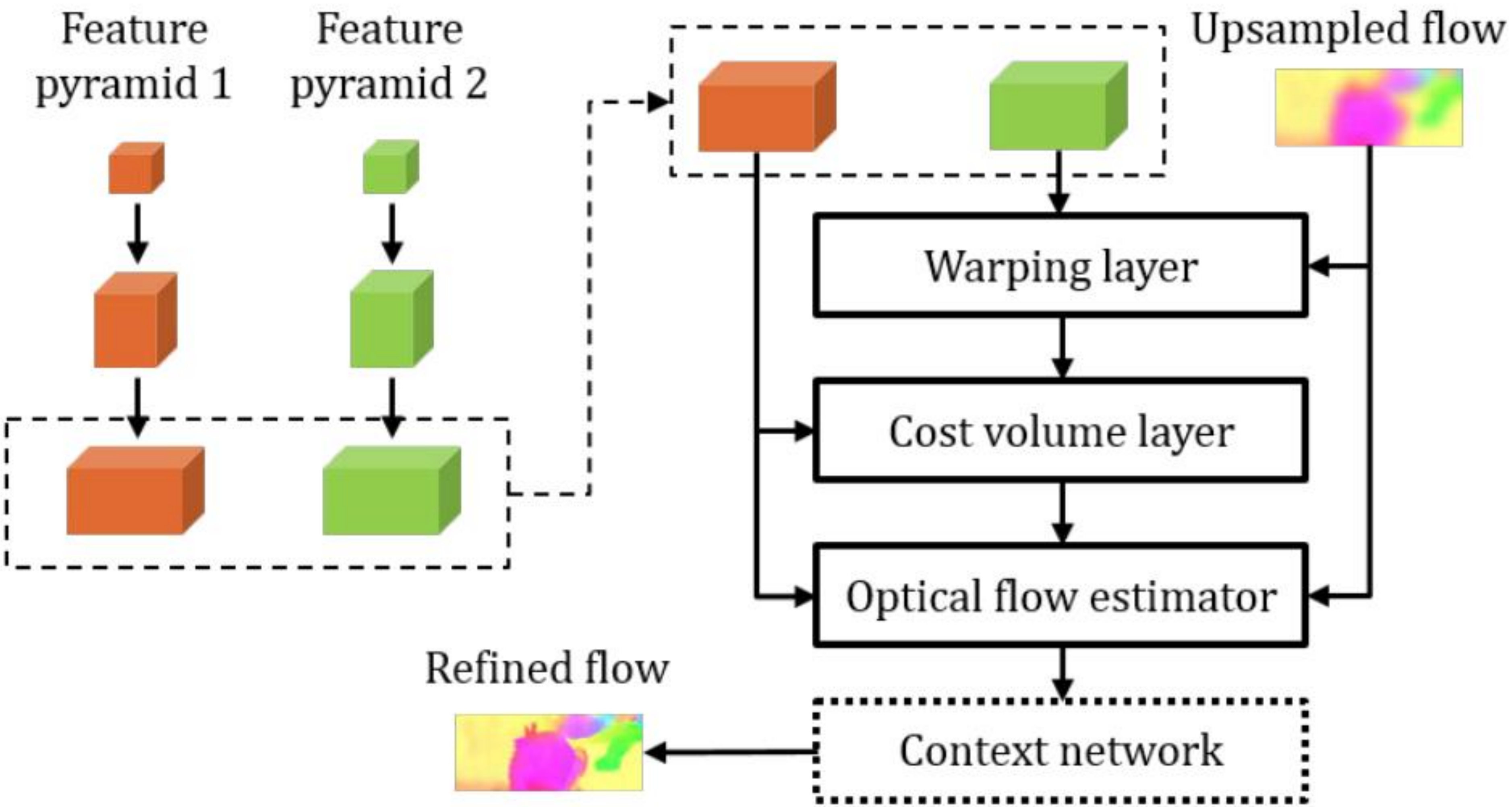




SpyNet (CVPR 2017)

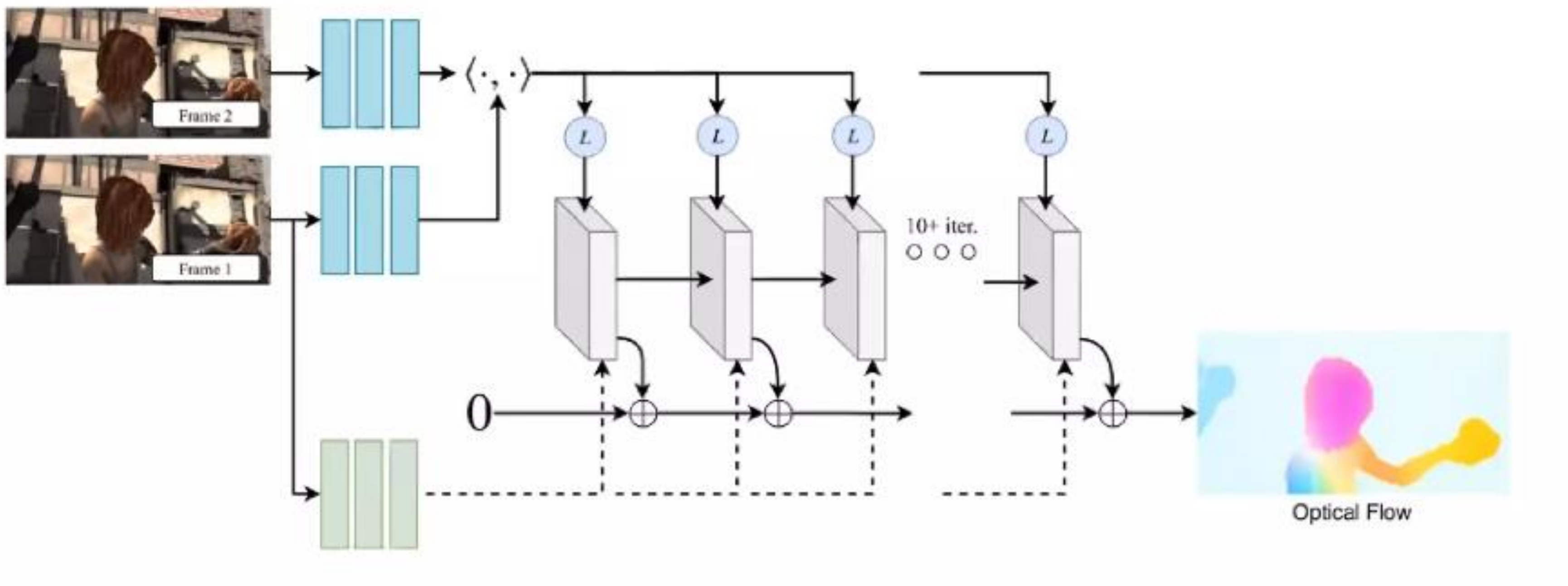


PWC-Net (CVPR 2018)



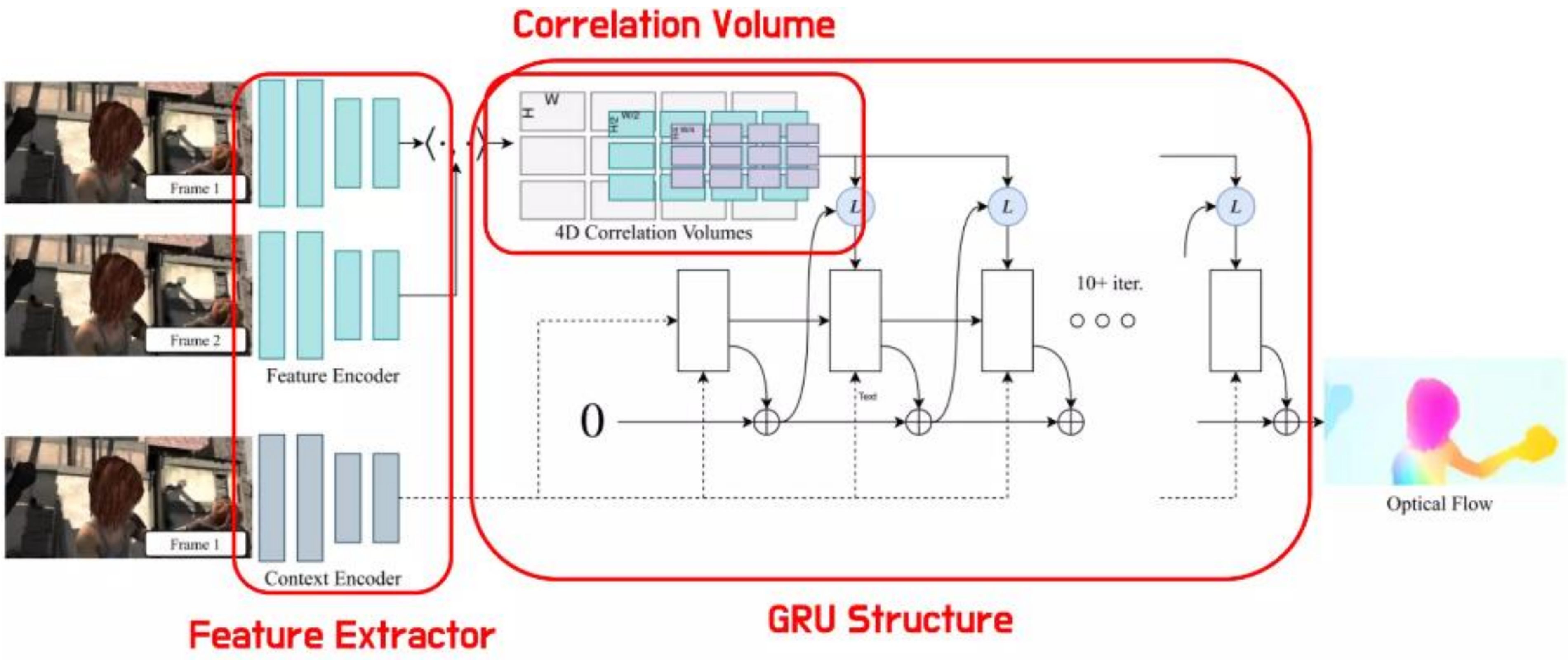
RAFT (ECCV 2020, Best Paper)

上海科技大学
ShanghaiTech University



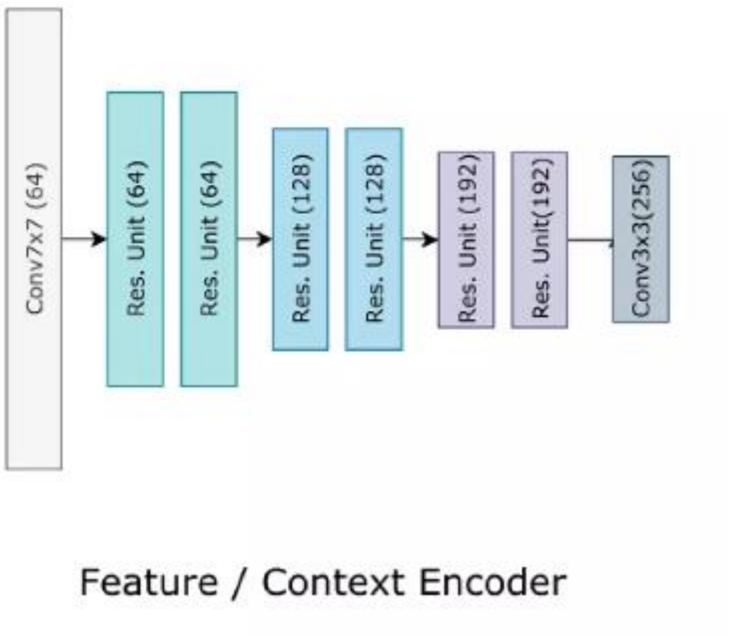
RAFT (ECCV 2020, Best Paper)

上海科技大学
ShanghaiTech University



Encoder

上海科技大学
ShanghaiTech University

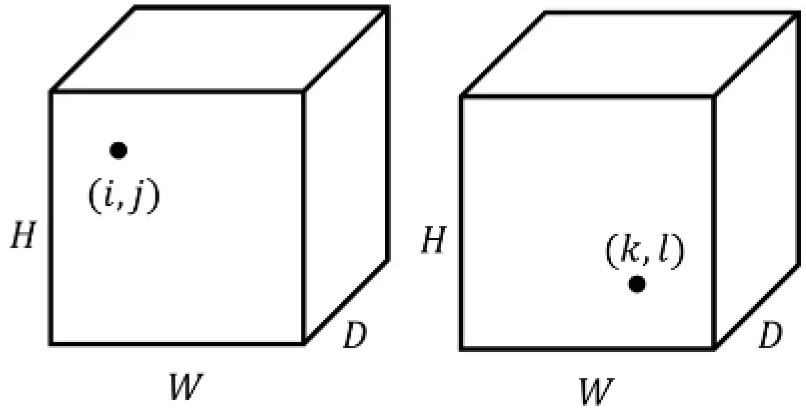


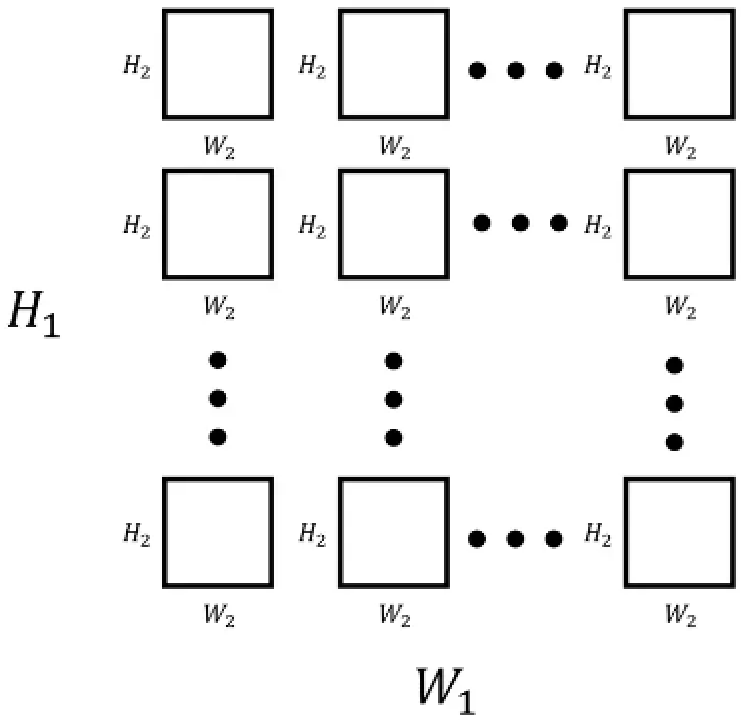
GRU – Cost Volume

Before Iteration:

$$\mathbf{C}(g_\theta(I_1), g_\theta(I_2)) \in \mathbb{R}^{H\times W\times H\times W}$$

$$C_{ijkl} = \sum_h g_\theta(I_1)_{ijh} \cdot g_\theta(I_2)_{klh}$$





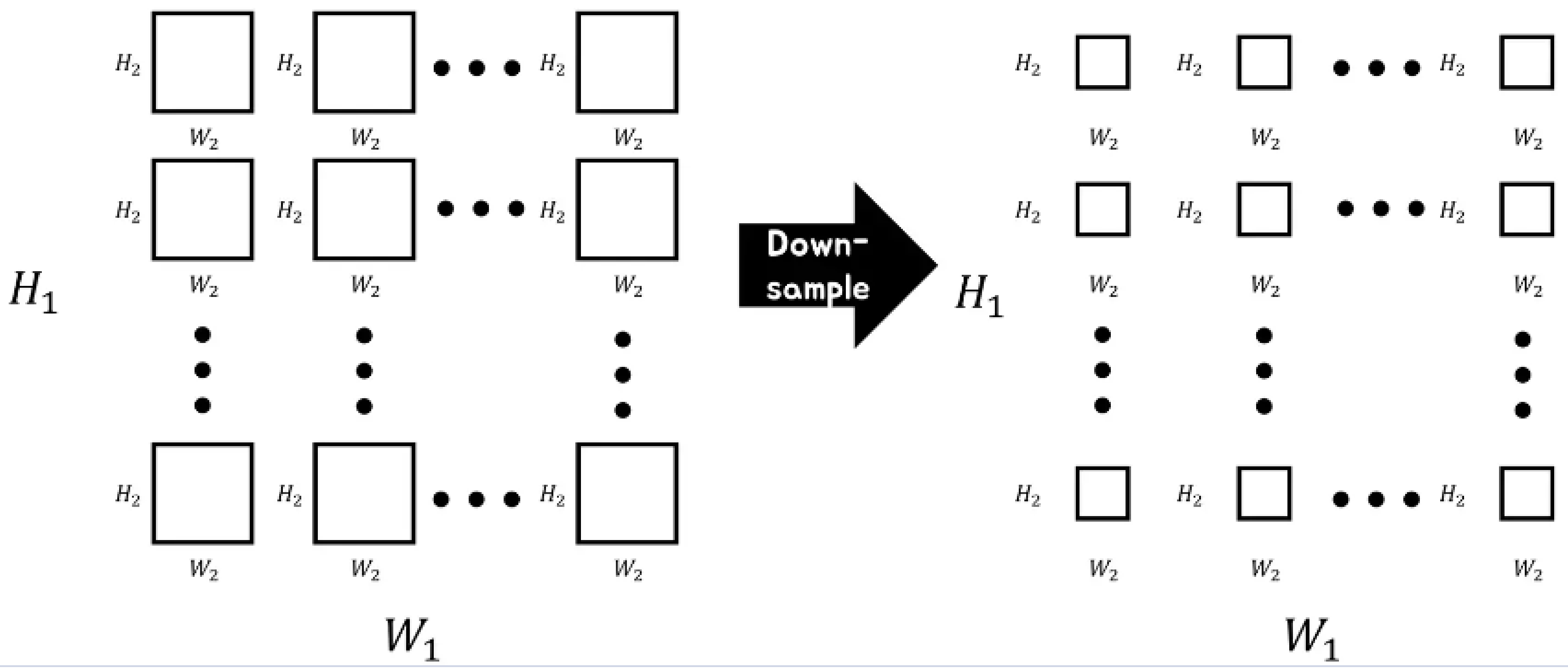
GRU -- Multi-scale Cost Volume

上海科技大学 ShanghaiTech University

Before Iteration:

- **Correlation Pyramid**

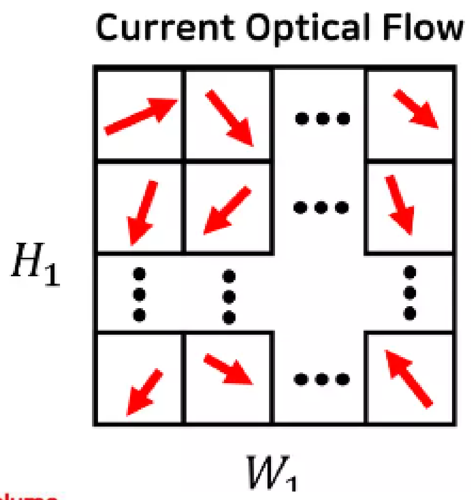
$$C^k: [H \times W \times \frac{H}{2^k} \times \frac{W}{2^k}]$$

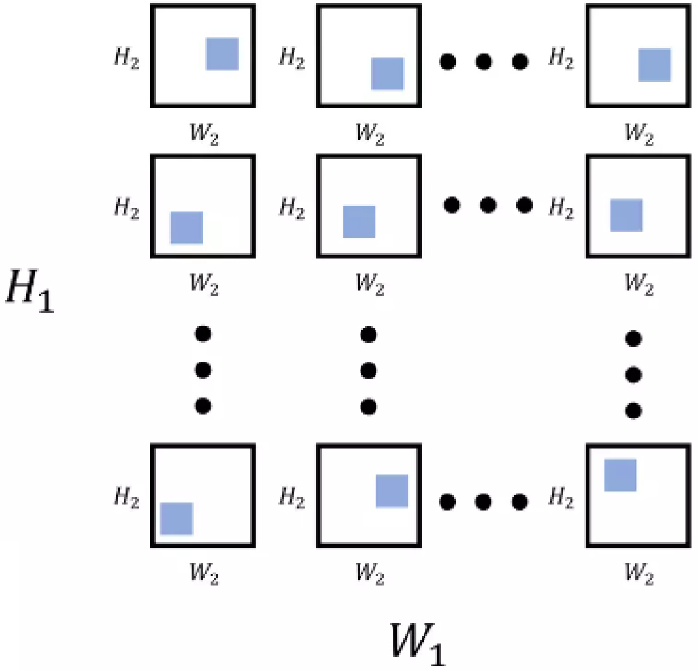


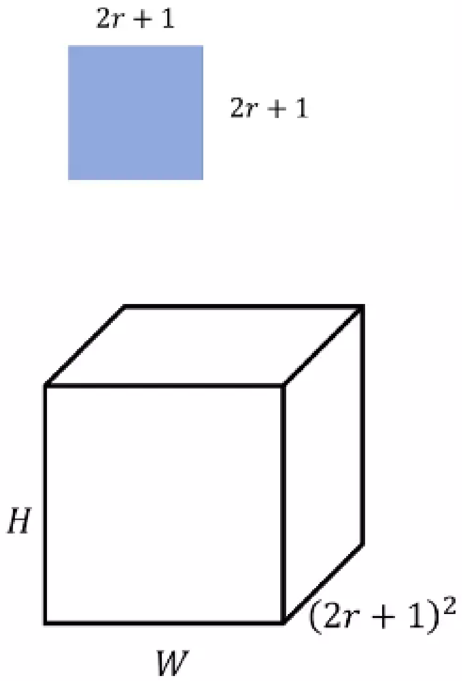
GRU – Cost Volume

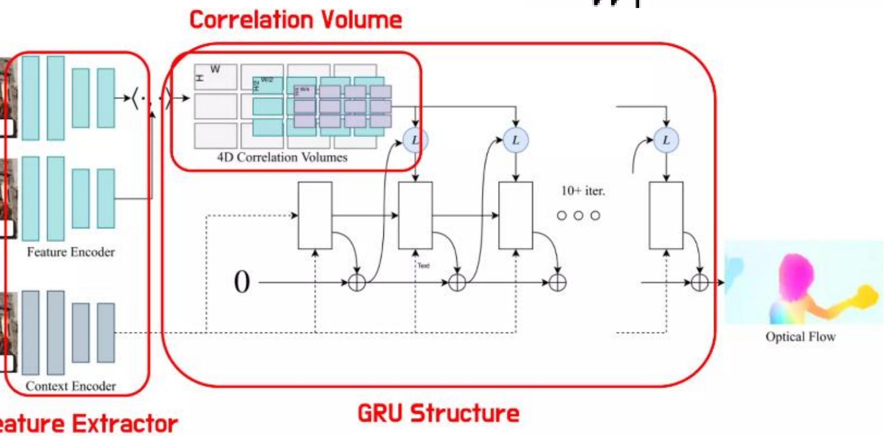
During Iteration:

- Correlation Lookup (4D Cost Volume → 3D Correlation Feature)







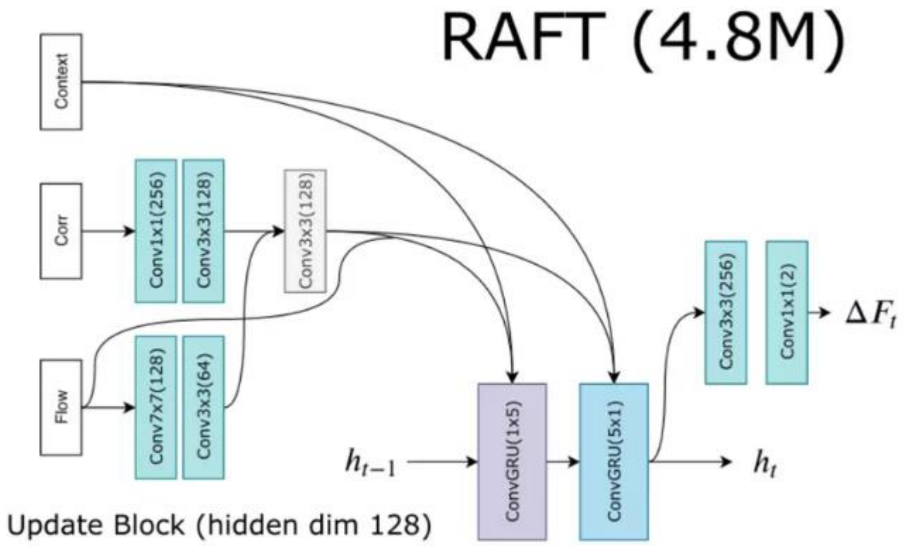


GRU – Iteration

During Iteration:

- h_t: Hidden Unit of GRU
- x_t: Flow, Correlation Feature, Context Feature

$$z_t = \sigma(\mathrm{Conv}_{3x3}([h_{t-1}, x_t], W_z))$$
$$r_t = \sigma(\mathrm{Conv}_{3x3}([h_{t-1}, x_t], W_r))$$
$$\tilde{h}_t = \tanh(\mathrm{Conv}_{3x3}([r_t \odot h_{t-1}, x_t], W_h))$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$



$$f_{k+1} = f_k + \Delta f$$

GRU – Upsampling