Deep Learning for various CV tasks

Computer Vision (SJK02)

Universitat Jaume I

Part A:

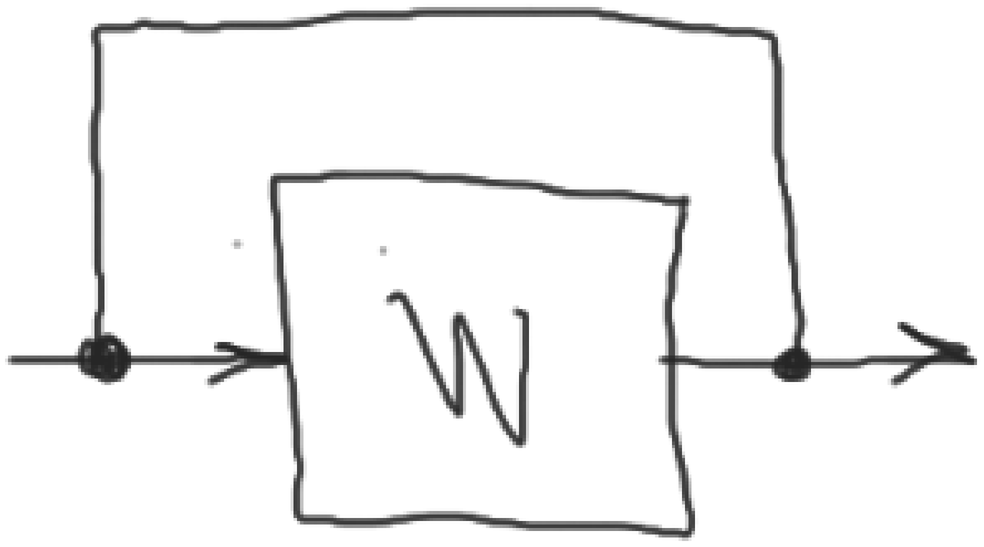
Classification
Segmentation
Object detection
(Image-based) biometrics

Part B:

Sequence processing
Optical flow
Action Recognition
Self-supervised learning
Transformers

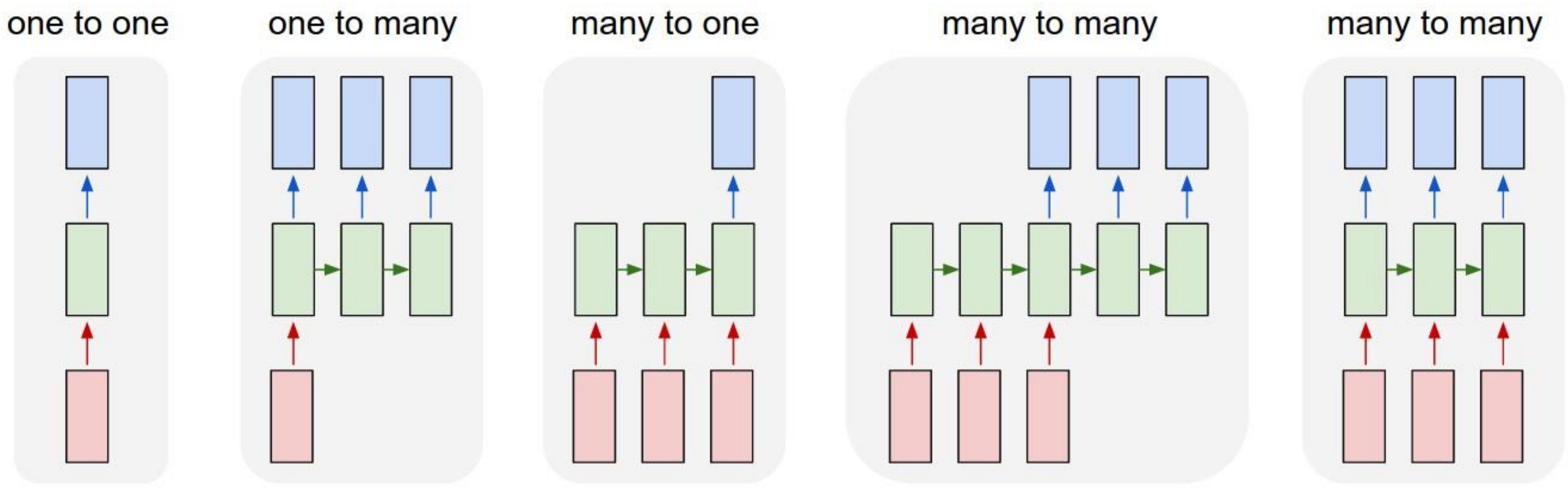
Sequence processing: RNNs, LSTMs

Recurrent Neural Networks



Unrolling

Flexible input/output sizes



Match problems to models

Frame-wise video classification

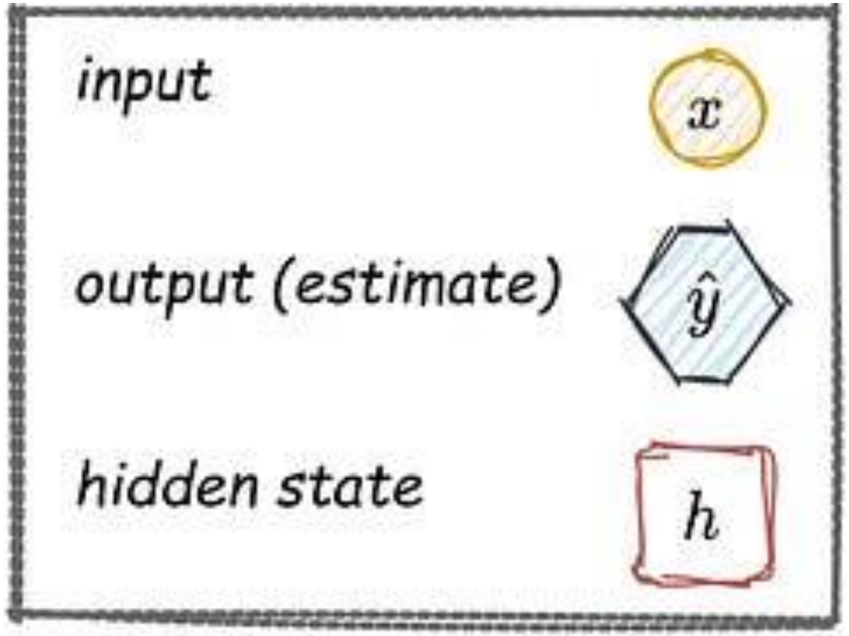
Image captioning

Image classification

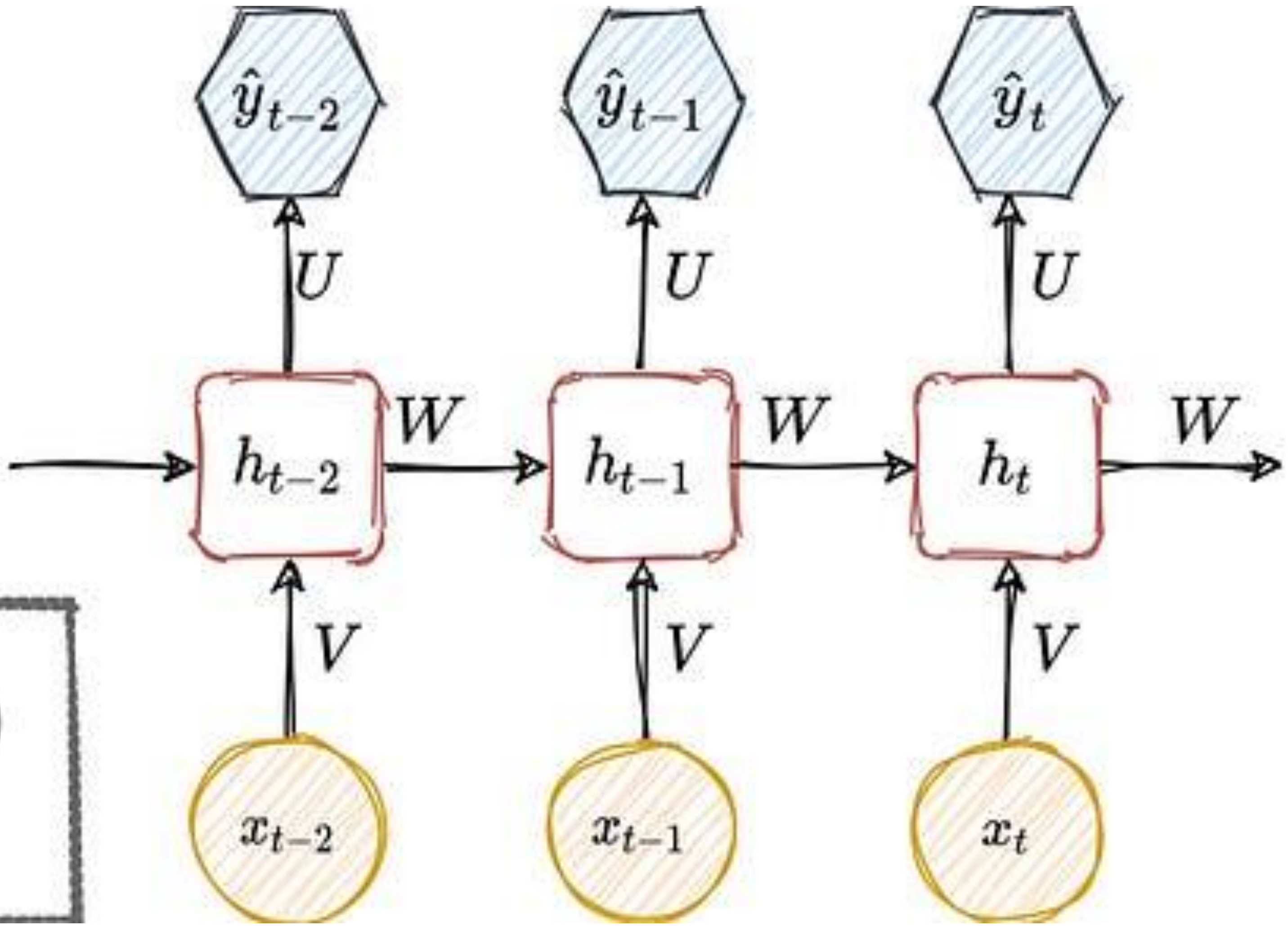
Sentiment analysis

Machine translation

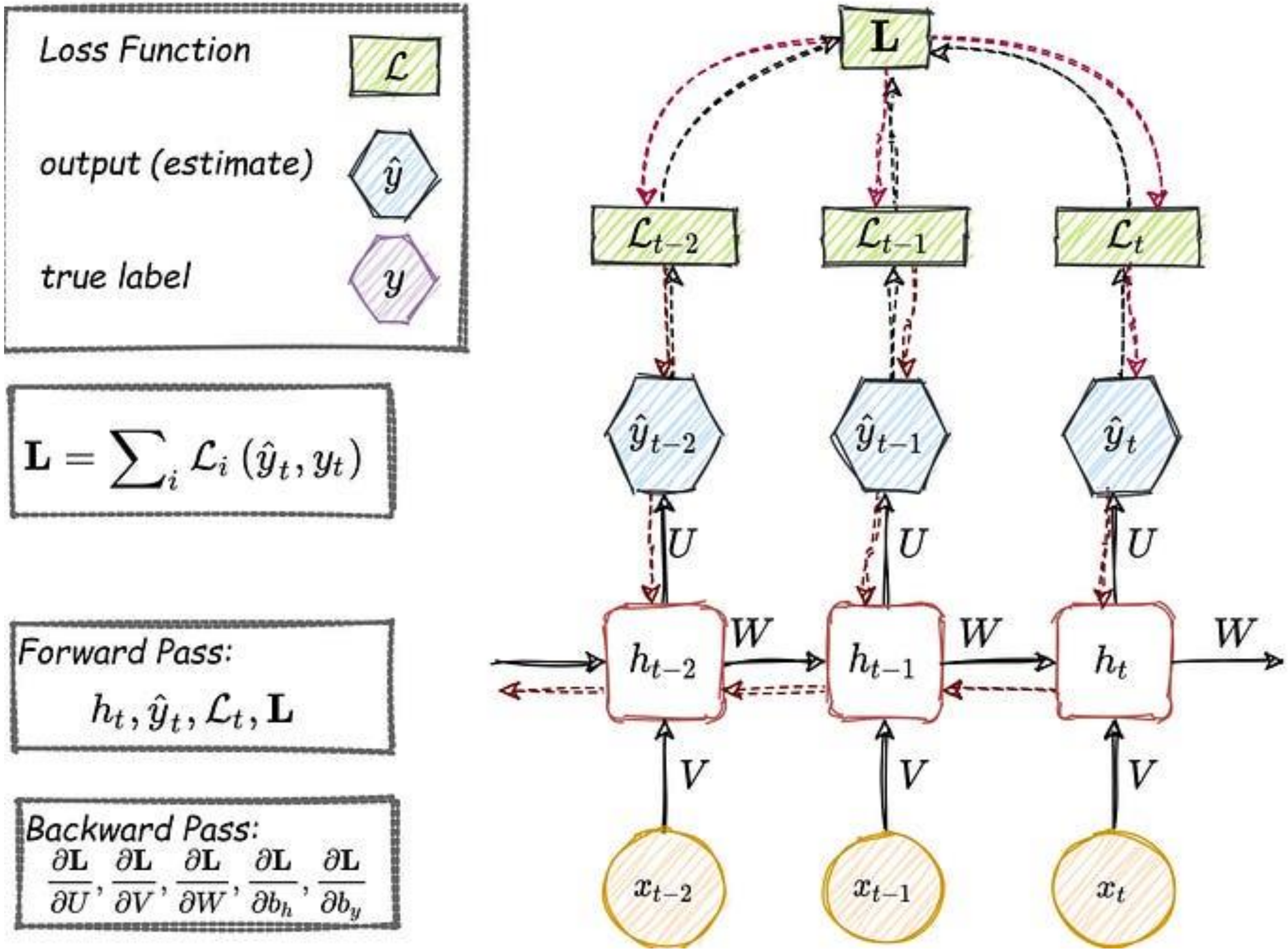
The Unreasonable Effectiveness of Recurrent Neural Networks (http://karpathy.github.io/2015/05/21/rnn-effectiveness)



$$h_t = f_h\left(Vx_t + Wh_{t-1} + b_h\right)$$
$$\hat{y}_t = f_y\left(Uh_t + b_y\right)$$



Backpropagation through time (BTT)



Exploding/vanishing gradients

$$\frac{\partial \mathbf{L}}{\partial W} \propto \sum_{i=0}^{T} \left(\prod_{i=k+1}^{y} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W}$$

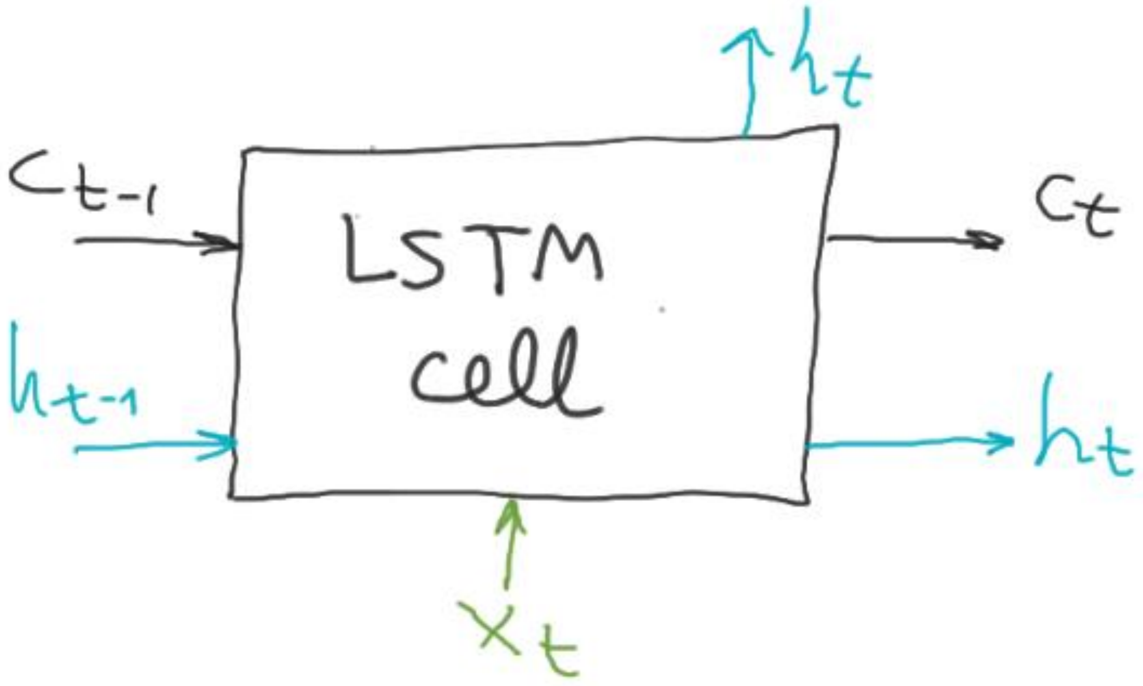
1. Vanishing gradient $\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 < 1$
2. Exploding gradient $\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 > 1$

https://towardsdatascience.com/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22

Dealing with this problem

- Skip connections (like ResNets)
- Remove 1-length connections
- Leaky recurrent units (regulate how much passed by α)
- Gated RN (learnable α)
- **LSTMs**

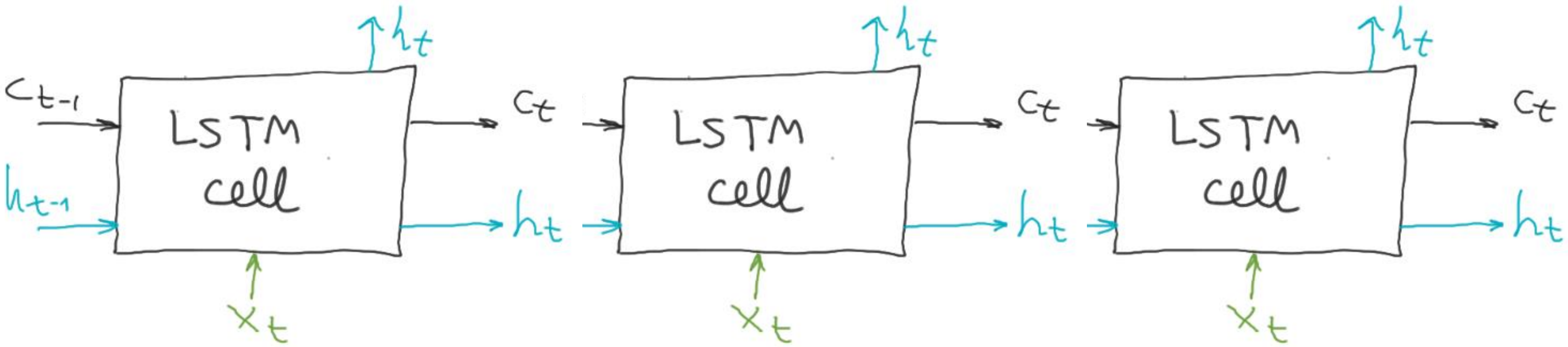
Long short-term memory (LSTM)



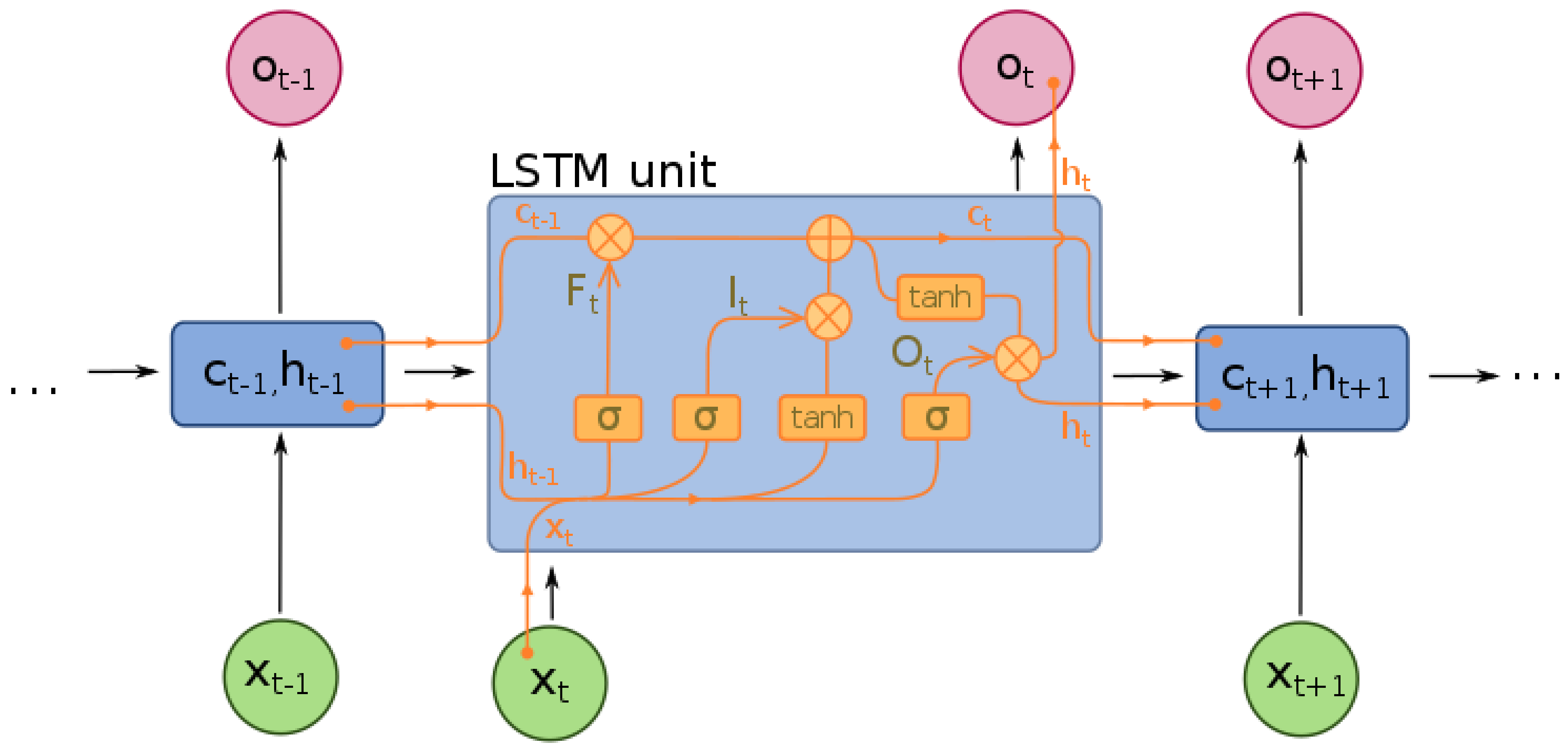
x_t = input
c_t = cell state ("memory")
h_t = hidden state (= output)

[Understanding LSTM Networks]

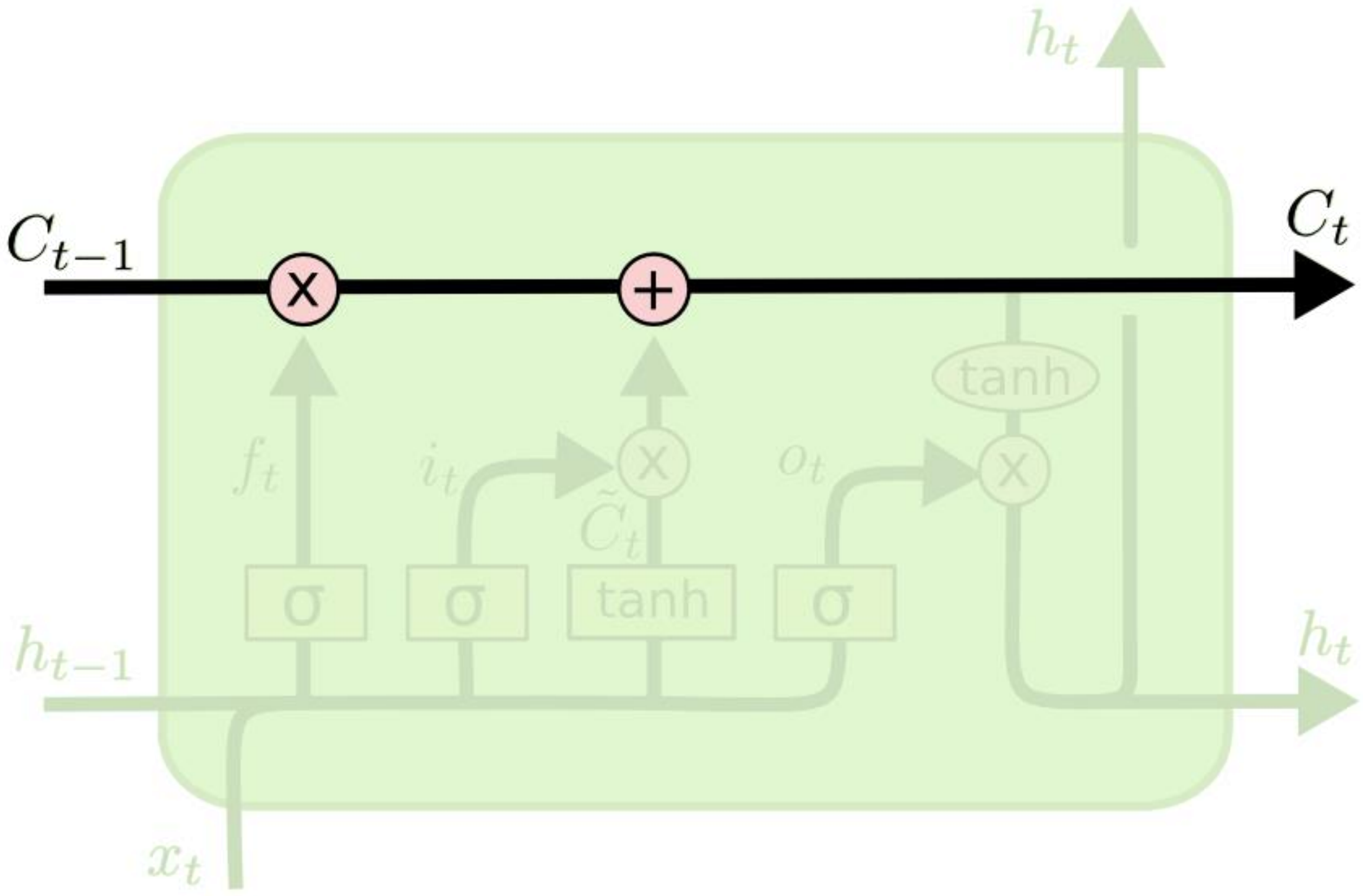
Unrolling



c_{t-1}
h_{t-1}
LSTM cell
LSTM cell
LSTM cell
h_t
h_t
h_t
c_t
c_t
c_t
h_t
h_t
h_t
x_t
x_t
x_t

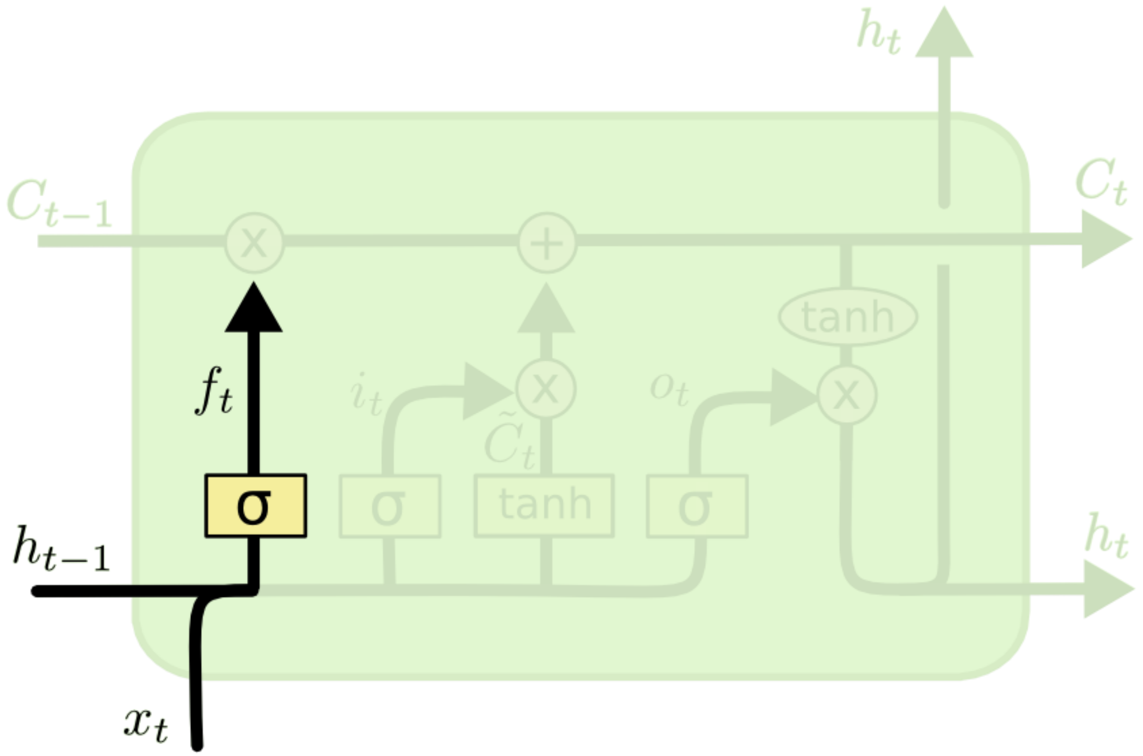


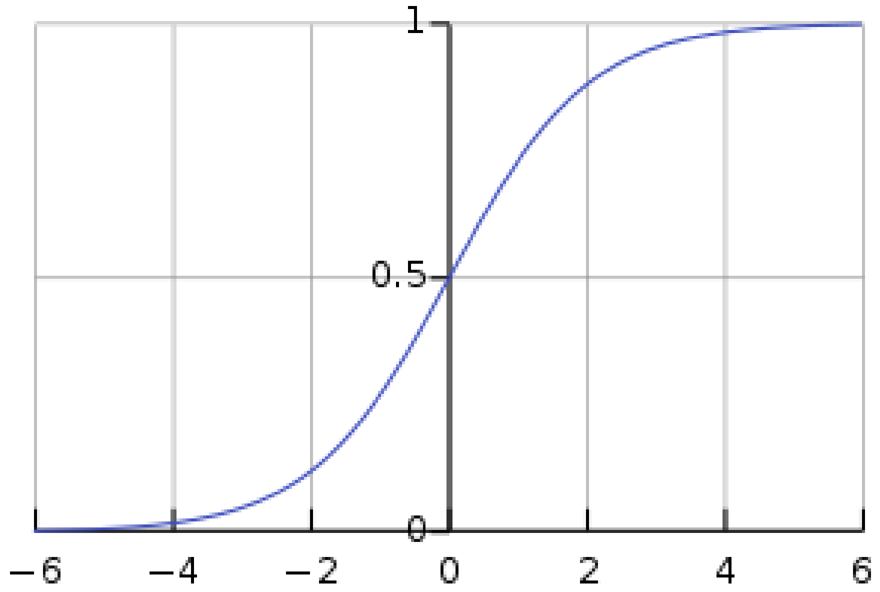
LSTM unit
o_{t-1}
o_t
o_{t+1}
c_{t-1}, h_{t-1}
c_{t+1}, h_{t+1}
x_{t-1}
x_t
x_{t+1}
c_{t-1}
c_t
h_t
h_{t-1}
F_t
I_t
O_t
σ
σ
tanh
σ
tanh



Cell **state** can go unchanged
Gates regulate how much it is changed

Forget gate



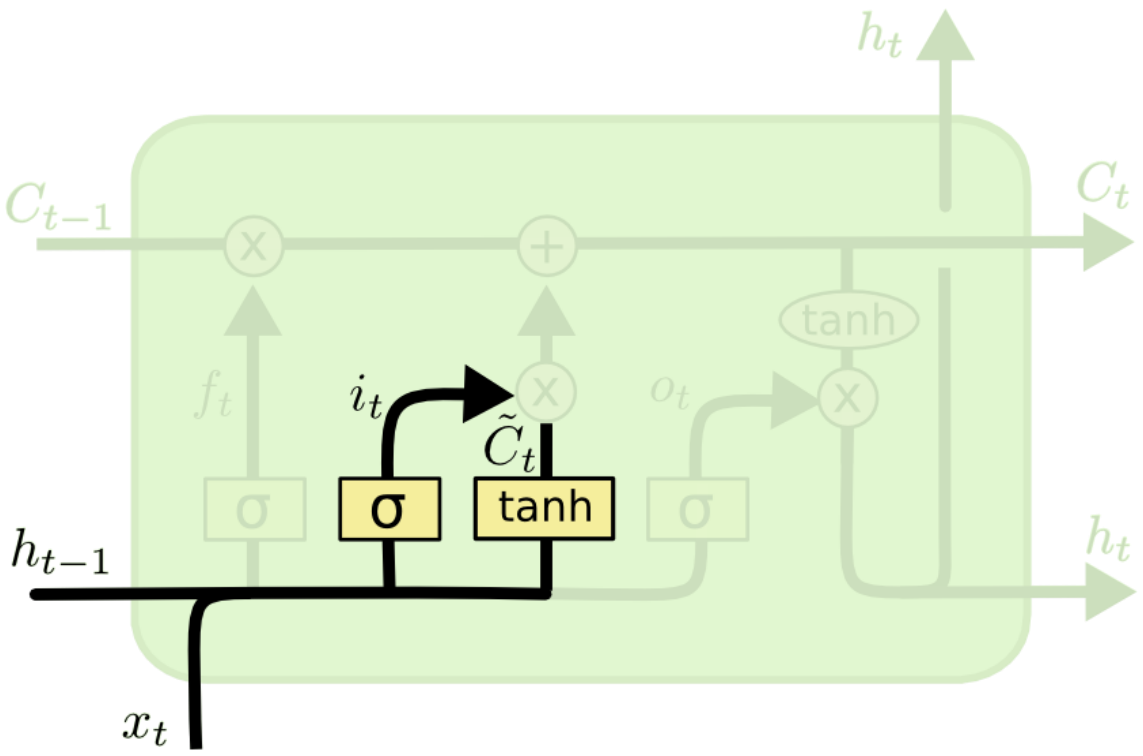


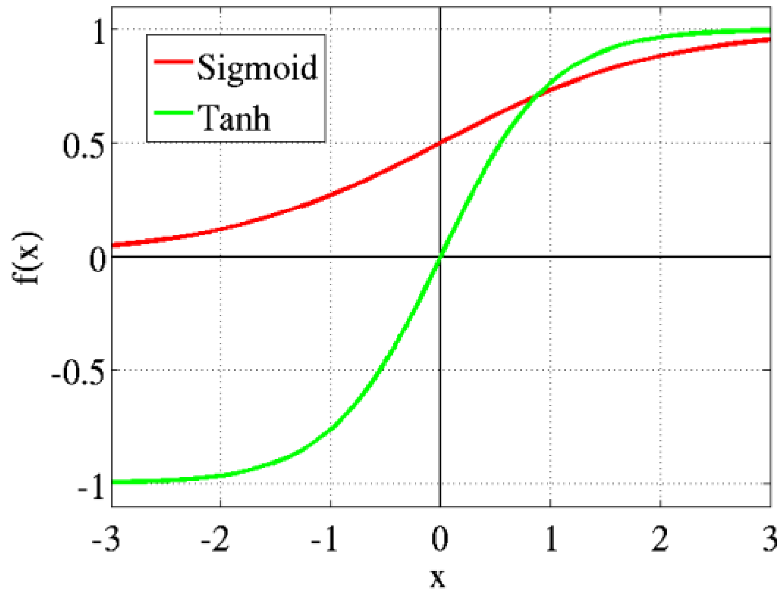
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x} = 1 - \sigma(-x).$$

How much, between 0 (nothing) and 1 (all), each element in C_{t-1} is kept

Input gate



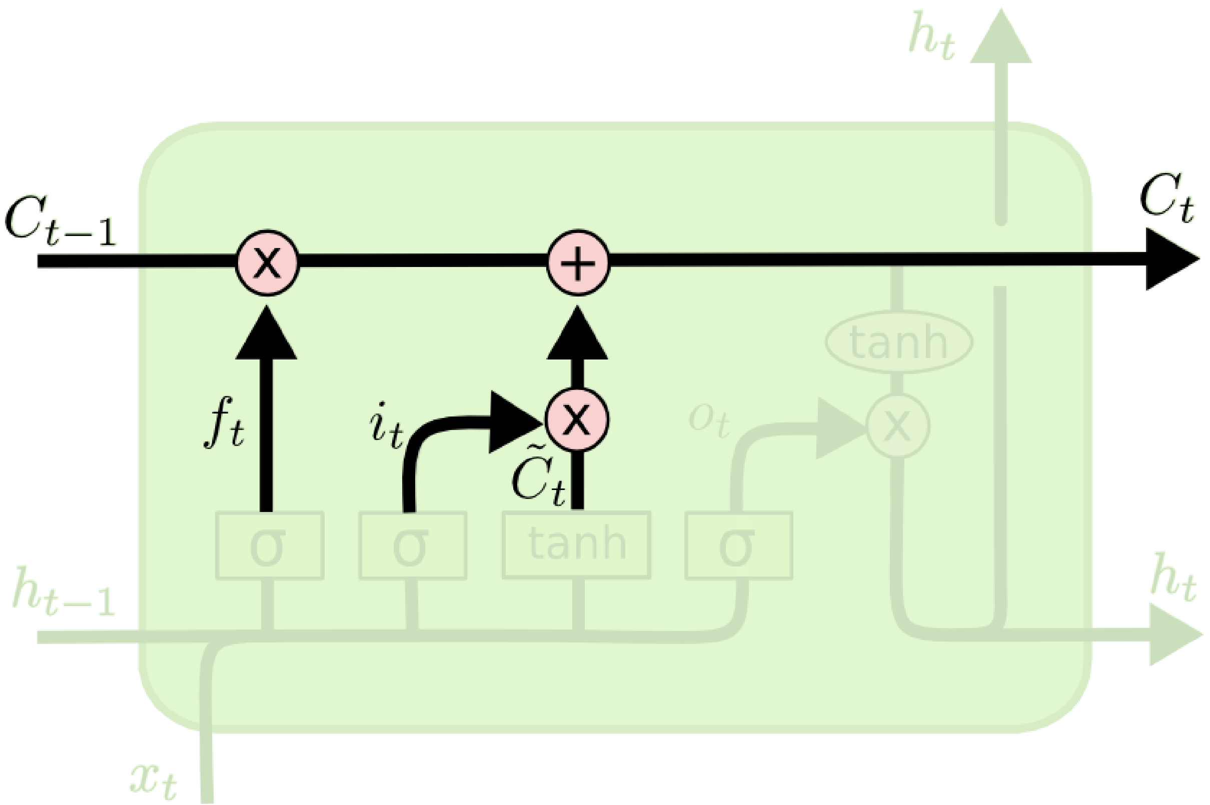


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

New information and how much of it to add to C_{t-1}

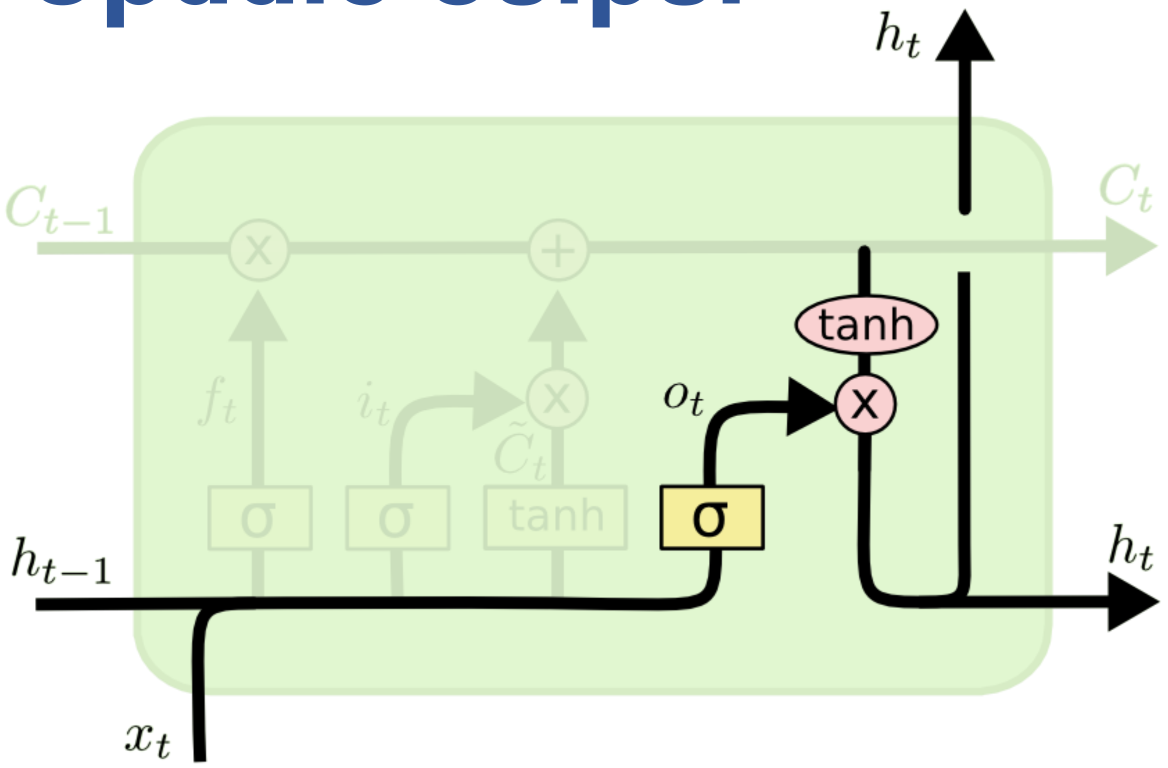
Update state



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

Update C_{t-1} to get C_t

Update output

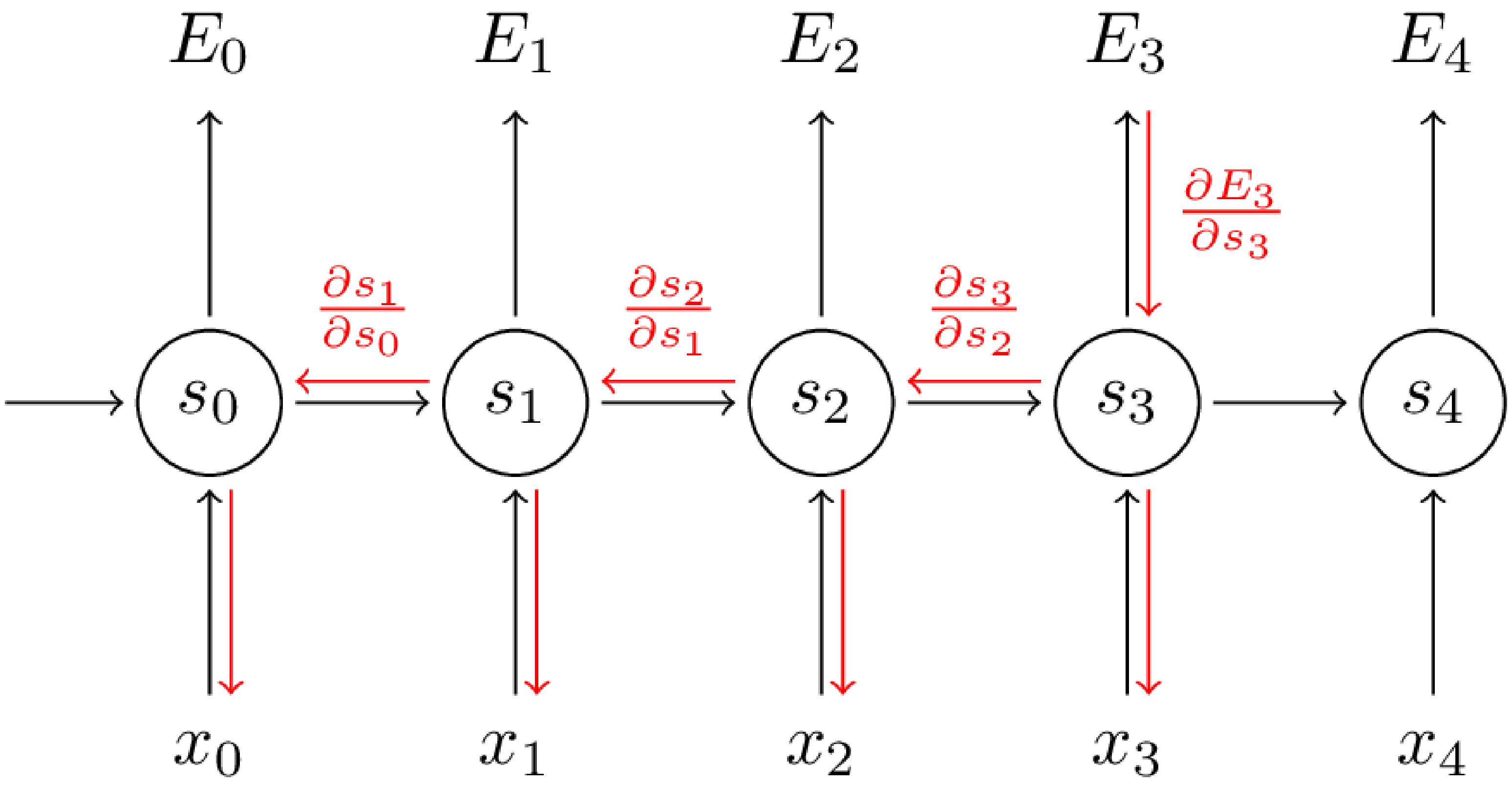


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

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

Update h_{t-1} to get h_t, a filtered version of the cell state C_t

Backpropagation through time (BTT)



[A Beginner's Guide on Recurrent Neural Networks with PyTorch]

Further possibilities

Bidirectional LSTMsStacked
LSTMsConvLSTMs
Attention
Transformers

LSTMs with PyTorch

```
lstm_layer = nn.LSTM(input_dim, hidden_dim, n_layers)
```

$x_t \in \mathbb{R}^{input\_dim}$

e.g. $[-3, 1, 2.5, 7, 0.17]$

$input\_dim = 5$

$h_t \in \mathbb{R}^{hidden\_dim}$

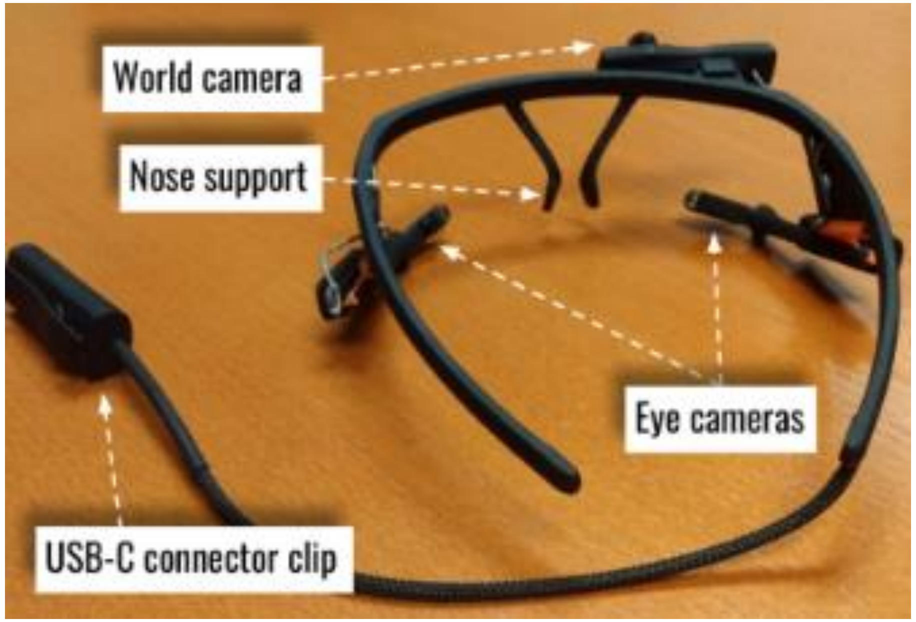
$c_t \in \mathbb{R}^{hidden\_dim}$

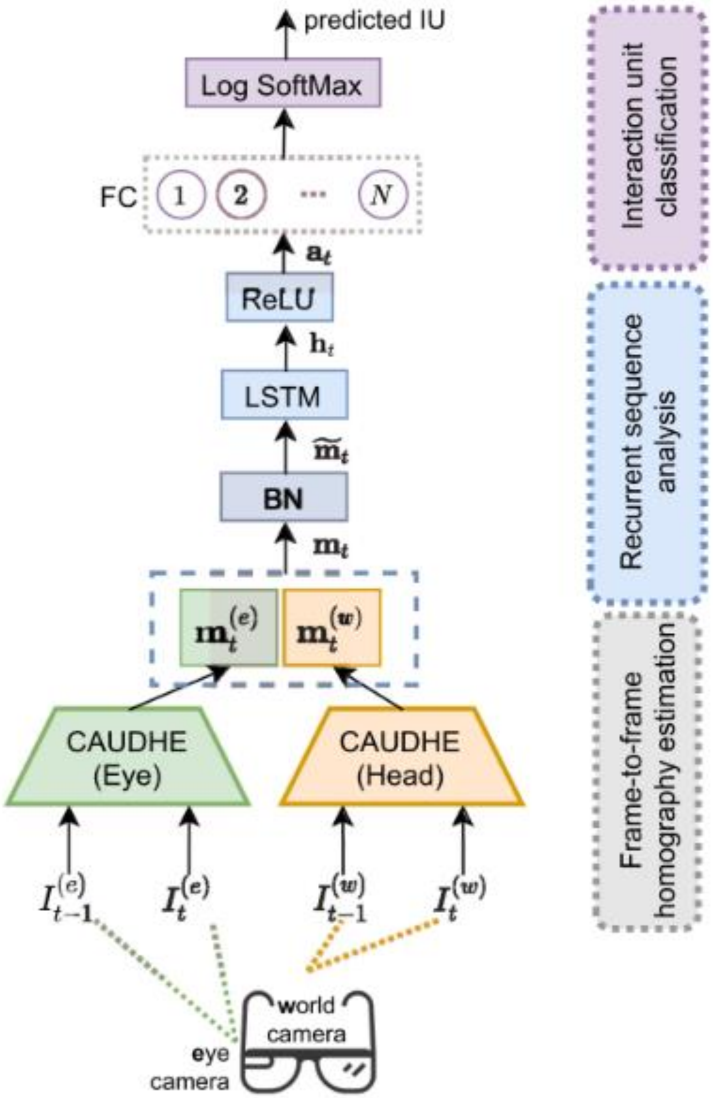
`n_layers = 1`
`(non-stacked LSTM)`

What about the sequence length?

[Long Short-Term Memory: From Zero to Hero with PyTorch]

Example: egocentric gesture recognition





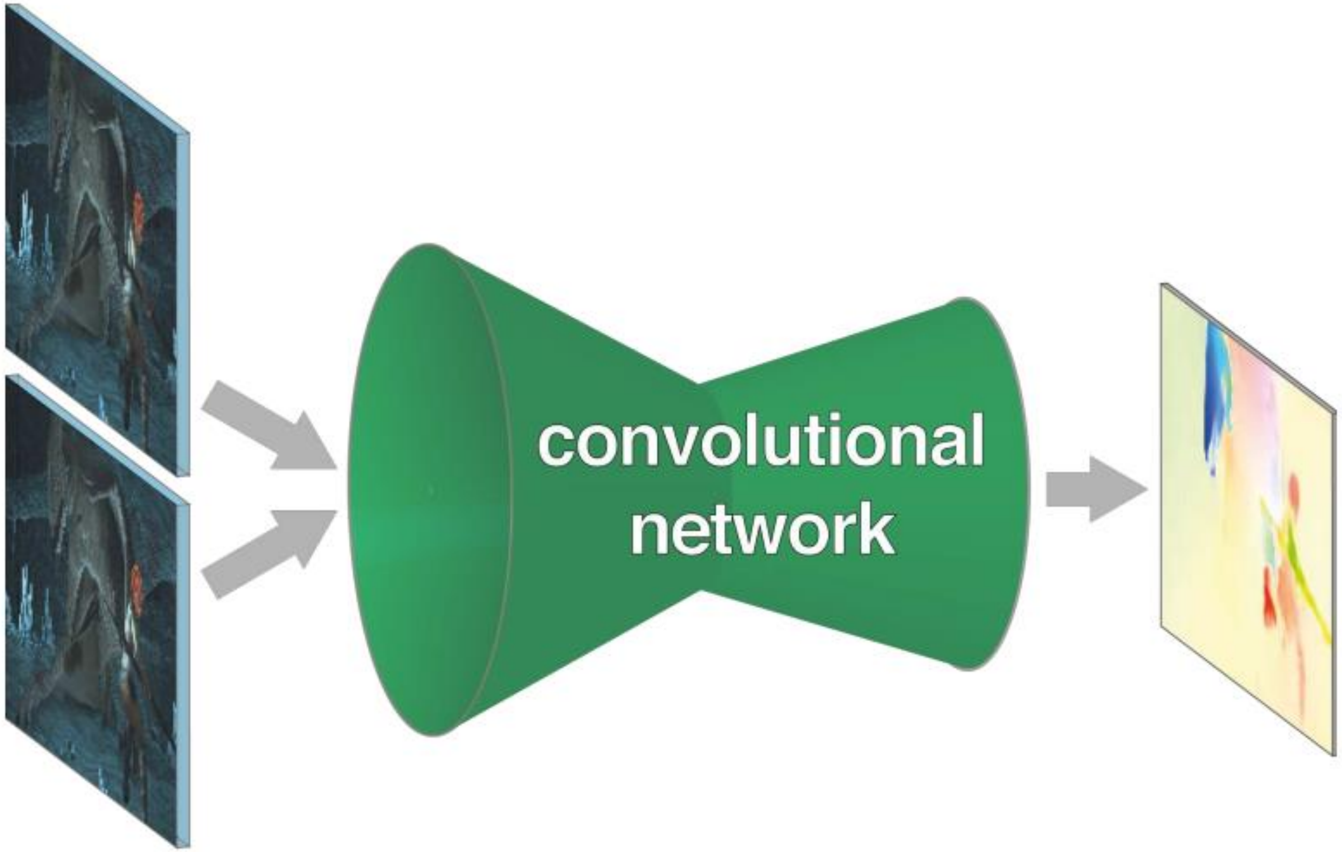
[Results (video)]

[Head and Eye Egocentric Gesture Recognition for Human-Robot Interaction Using Eyewear Cameras (IEEE Rob. & Aut. Letters, 2022)]

Optical flow:
FlowNet, SpyNet, LikeNet

FlowNet (ICCV2015)

Can optic flow estimation be *learned*?



[FlowNet: Learning Optical Flow with Convolutional Networks (ICCV 2015)]

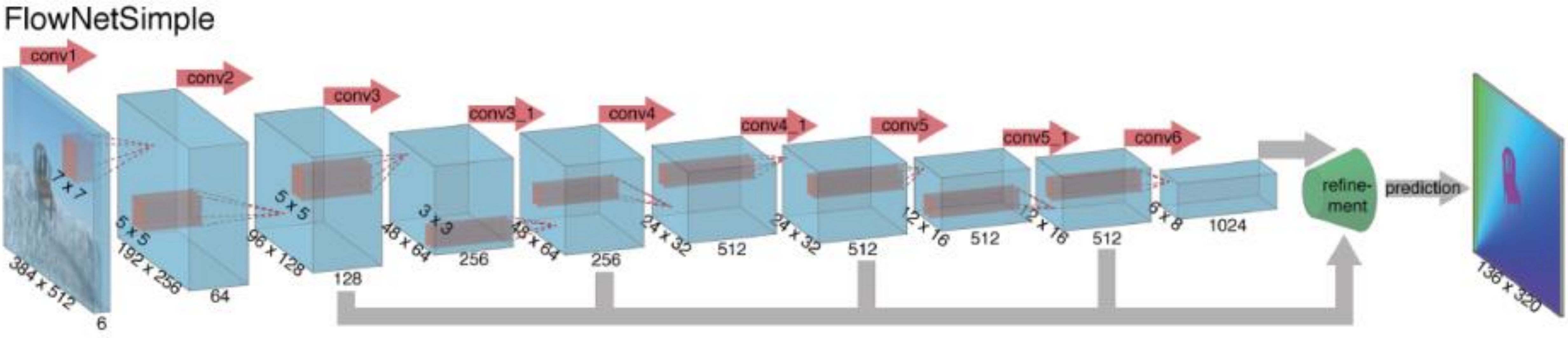
Supervised learning: what about the *labelled* data?

FlyingChairs

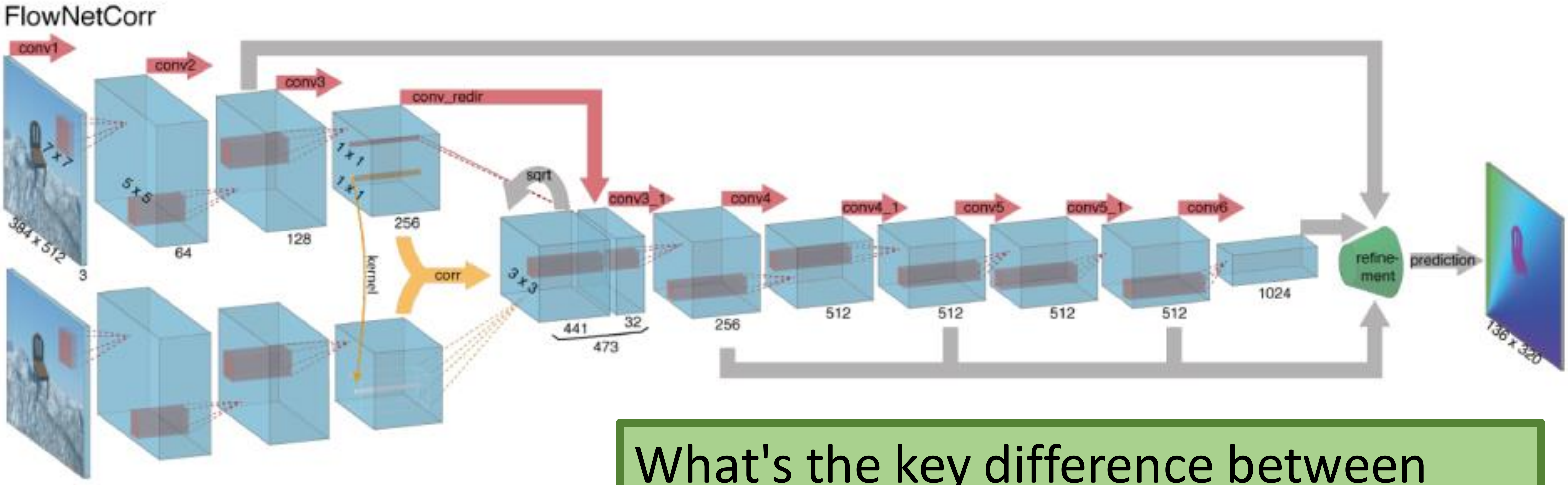
FlyingThings3D

[Sintel]

[FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks (CVPR 2017)]



[FlowNet: Learning Optical Flow with Convolutional Networks (ICCV 2015)]



What's the key difference between FlowNetSimple and FlowNetCorr?

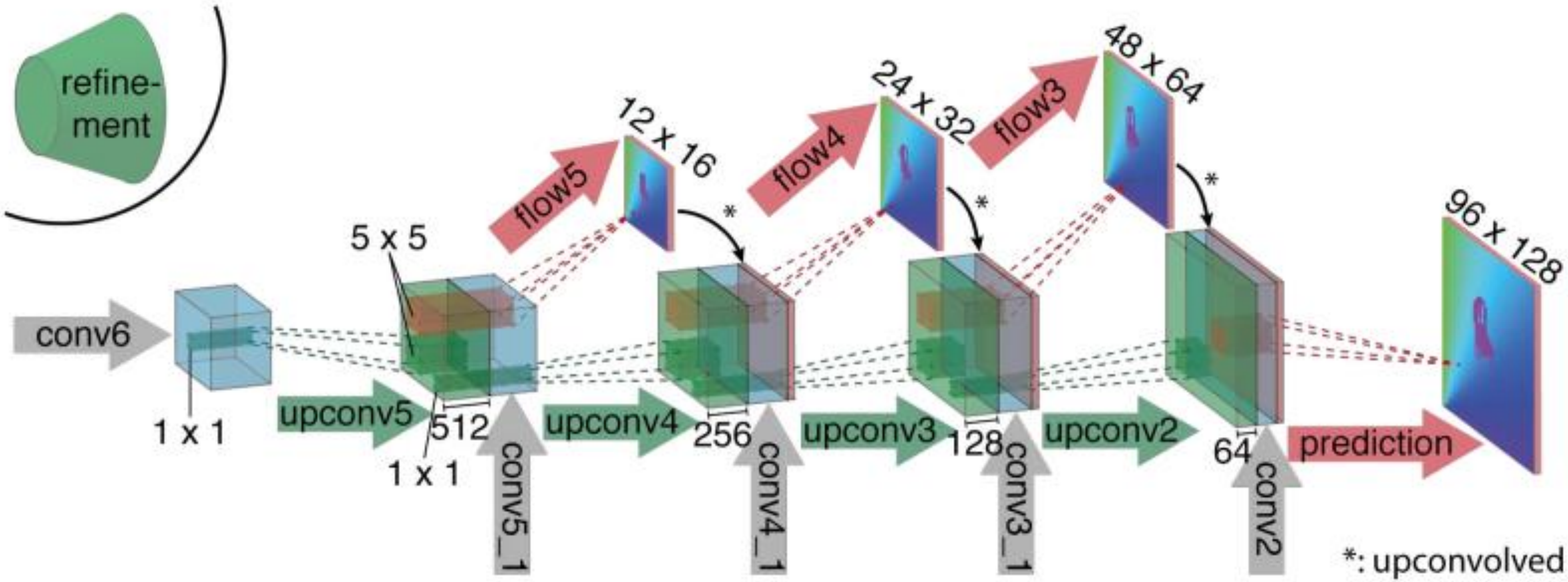
Correlation layer

$$c(\mathbf{x}_1, \mathbf{x}_2) = \sum_{\mathbf{o} \in [-k,k] \times [-k,k]} \langle \mathbf{f}_1(\mathbf{x}_1 + \mathbf{o}), \mathbf{f}_2(\mathbf{x}_2 + \mathbf{o}) \rangle$$

Like convolution between feature maps (no learnable weights)

Refinement

Upsampling



Findings

It is **possible to learn** to estimate optic flow

Training data need **not** be **realistic**

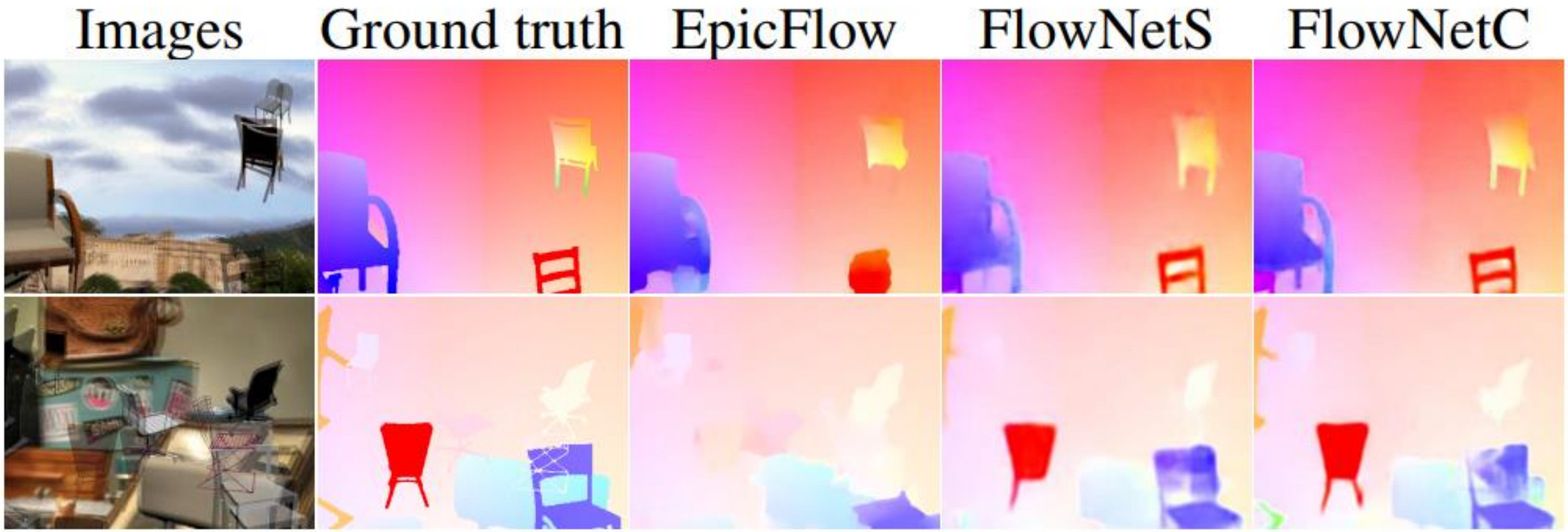
FlowNetSimple vs FlowNetCorr? --> Depends on dataset

FlowNetCorr:

- slighlty overfits the training data
- has some problems with large displacements

Why do you think authors relate the limitation of FlowNetCorr with large displacements with the correlation layer?

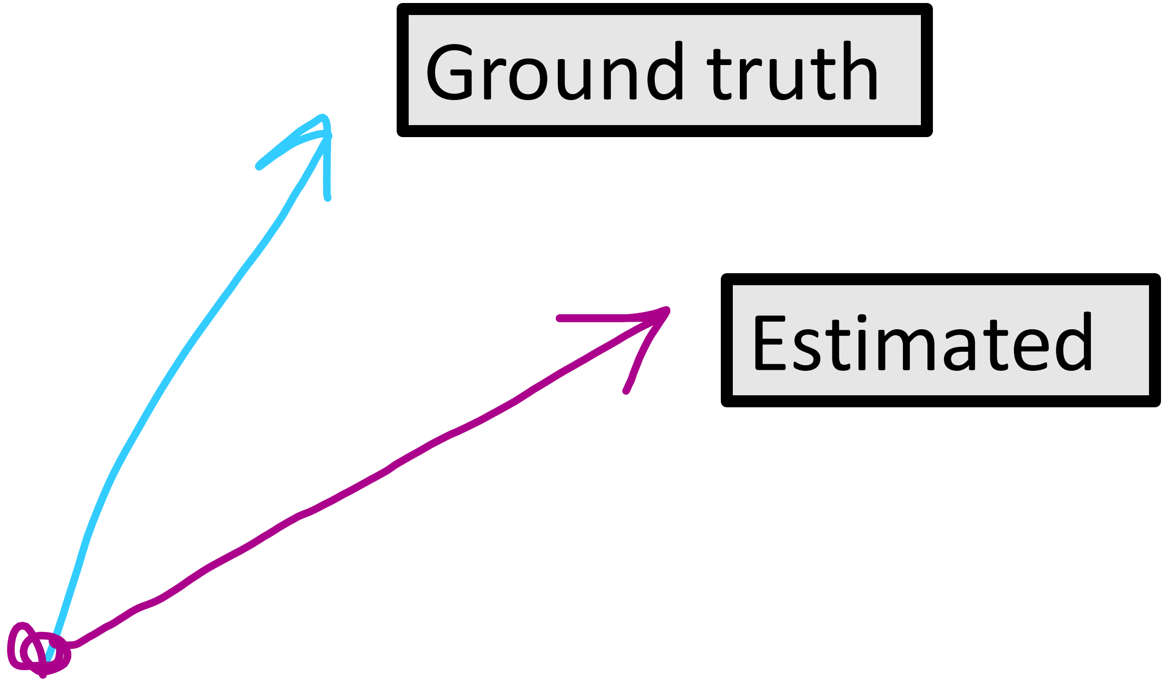
Some results



Find where FlowNetS is better than EpicFlow (CVPR 2015)
Find where FlowNetC is better than FlowNetS

Loss

End-point-error (EPE)



FlowNet 2.0 (CVPR 2017)

FlowNet 2.0 vs FlowNet (ICCV 2015)

- schedule of presenting data
- stacked architecture with warping
- subnetwork specializing on small motions

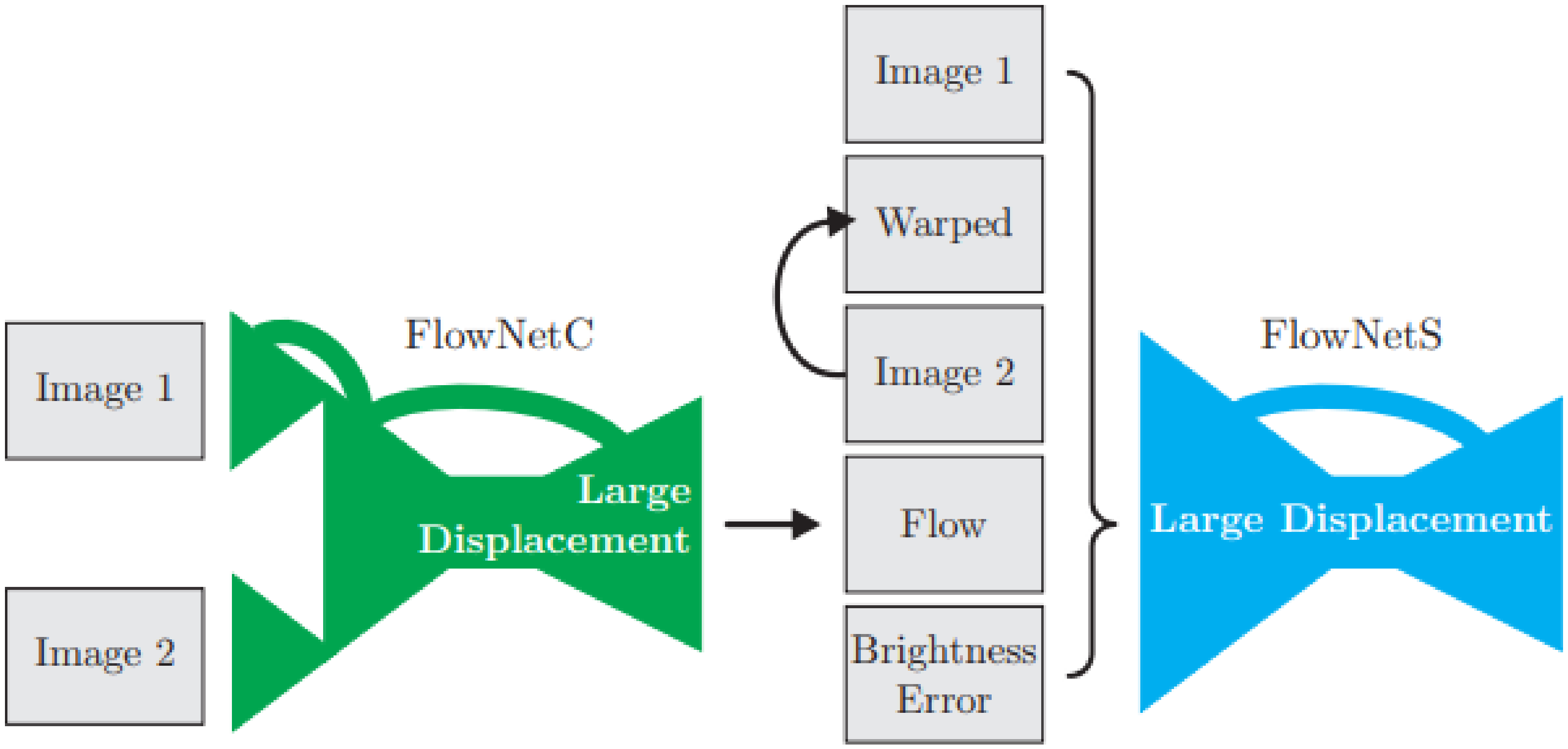
Data presentation is important

- Training only on Things3D (more realistic) --> worse results
- Best results: training on FlyingChairs first, then on Things3D

What would be a "take-home lesson" for other problems?

A form of curriculum learning?

Does iteration and warping help?



Stacking may help;
stacking+warping *always* help

Final net

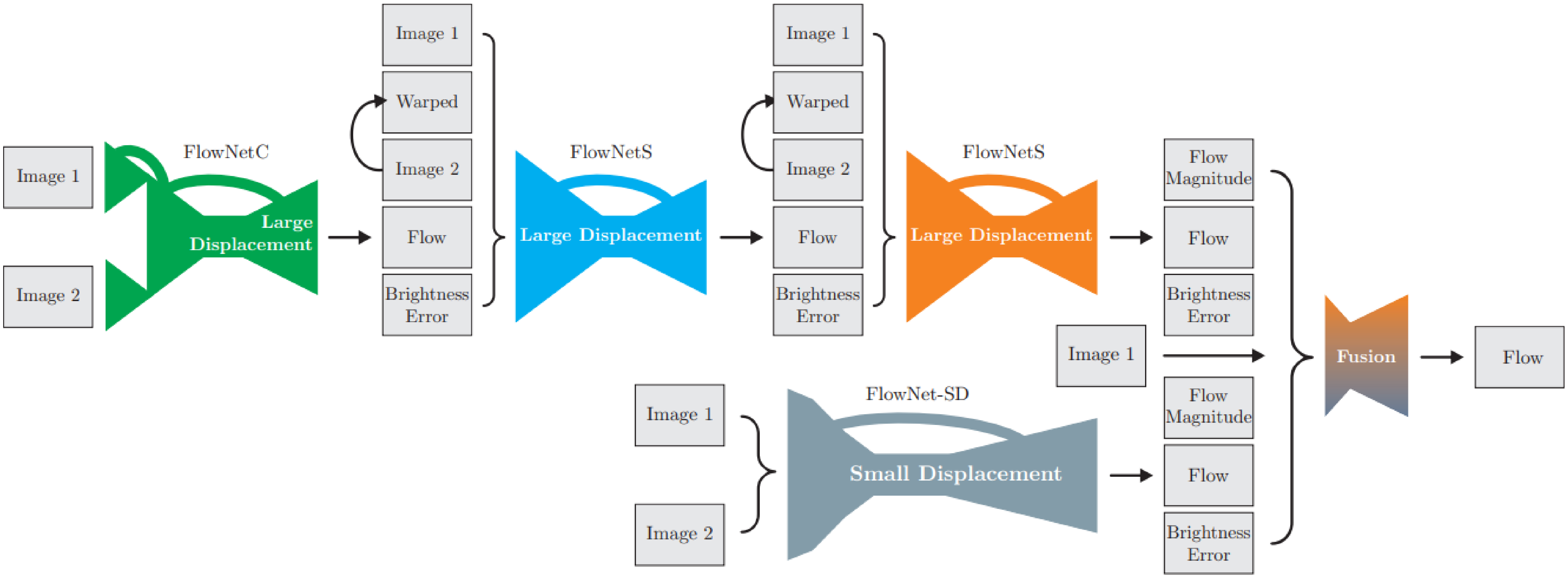
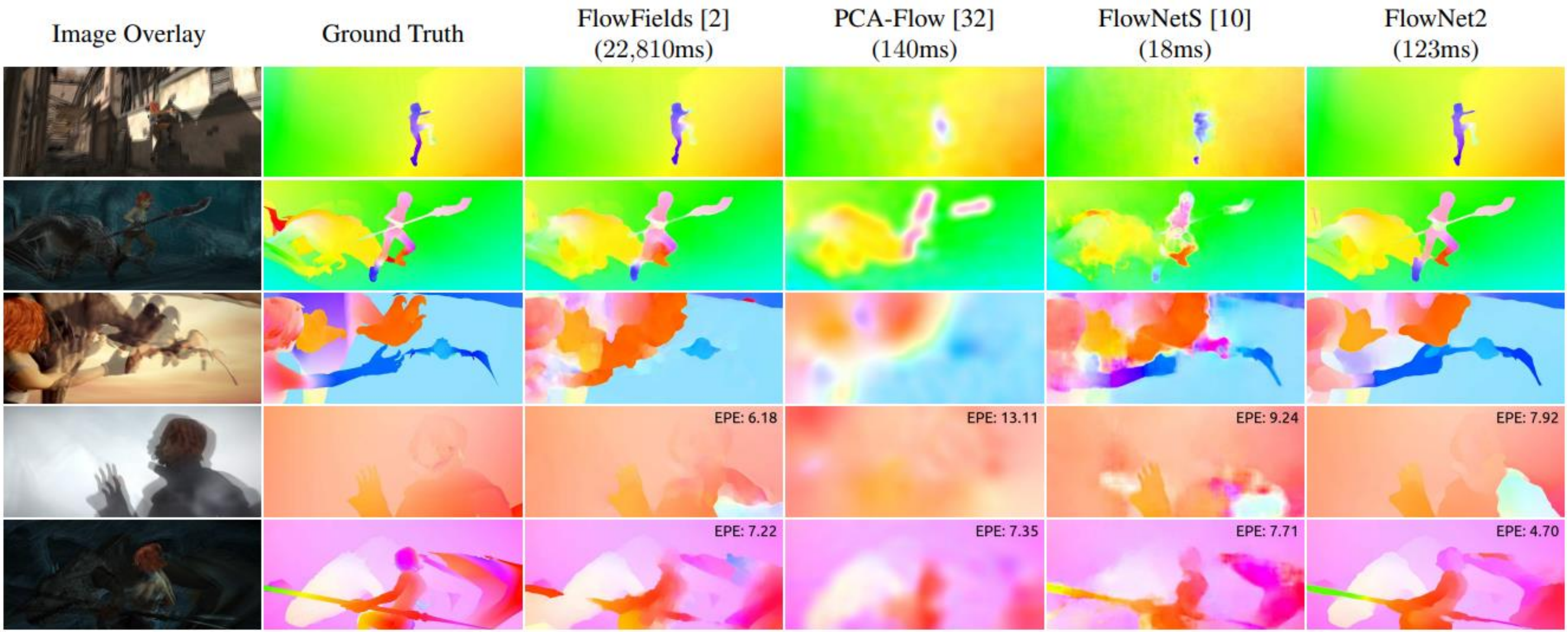
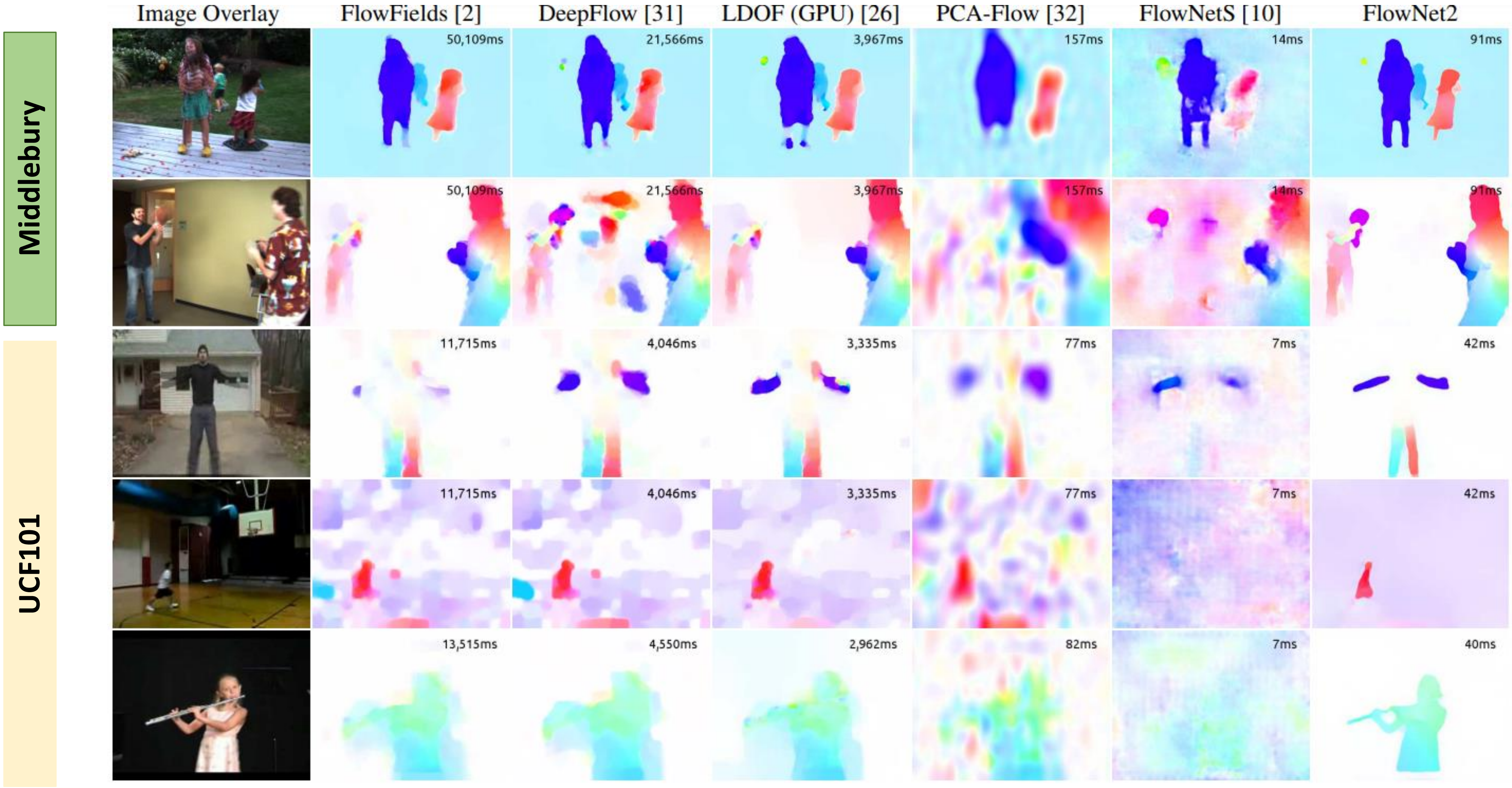


Image 1
Image 2
FlowNetC
Large
Displacement
Image 1
Warped
Image 2
Flow
Brightness
Error
FlowNetS
Large Displacement
Image 1
Warped
Image 2
Flow
Brightness
Error
FlowNetS
Large Displacement
Flow
Magnitude
Flow
Brightness
Error
Image 1
Image 1
Image 2
FlowNet-SD
Small Displacement
Flow
Magnitude
Flow
Brightness
Error
Fusion
Flow

Some results on Sintel



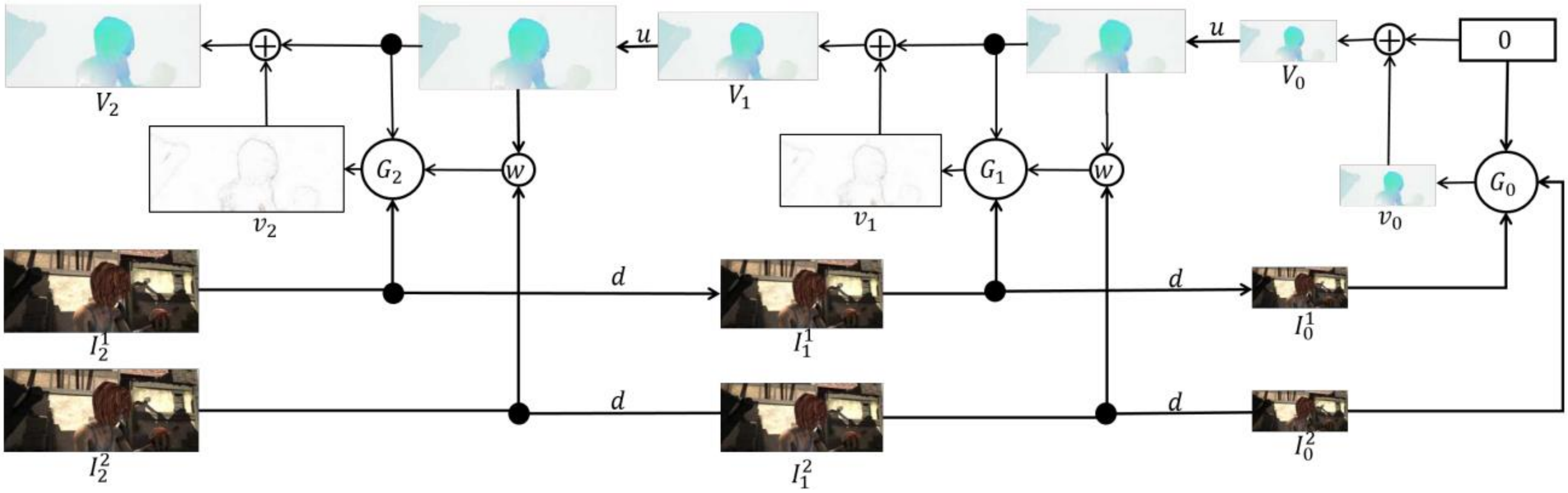
Some results on real data



Conclusions

- FlowNet 2.0 is marginally slower than FlowNet
- Estimation error reduced by more than 50%
- Performs on par with state-of-the-art methods
- Runs at interactive frame rates
- Faster variants run up to 140fps

SPyNet (CVPR 2017)



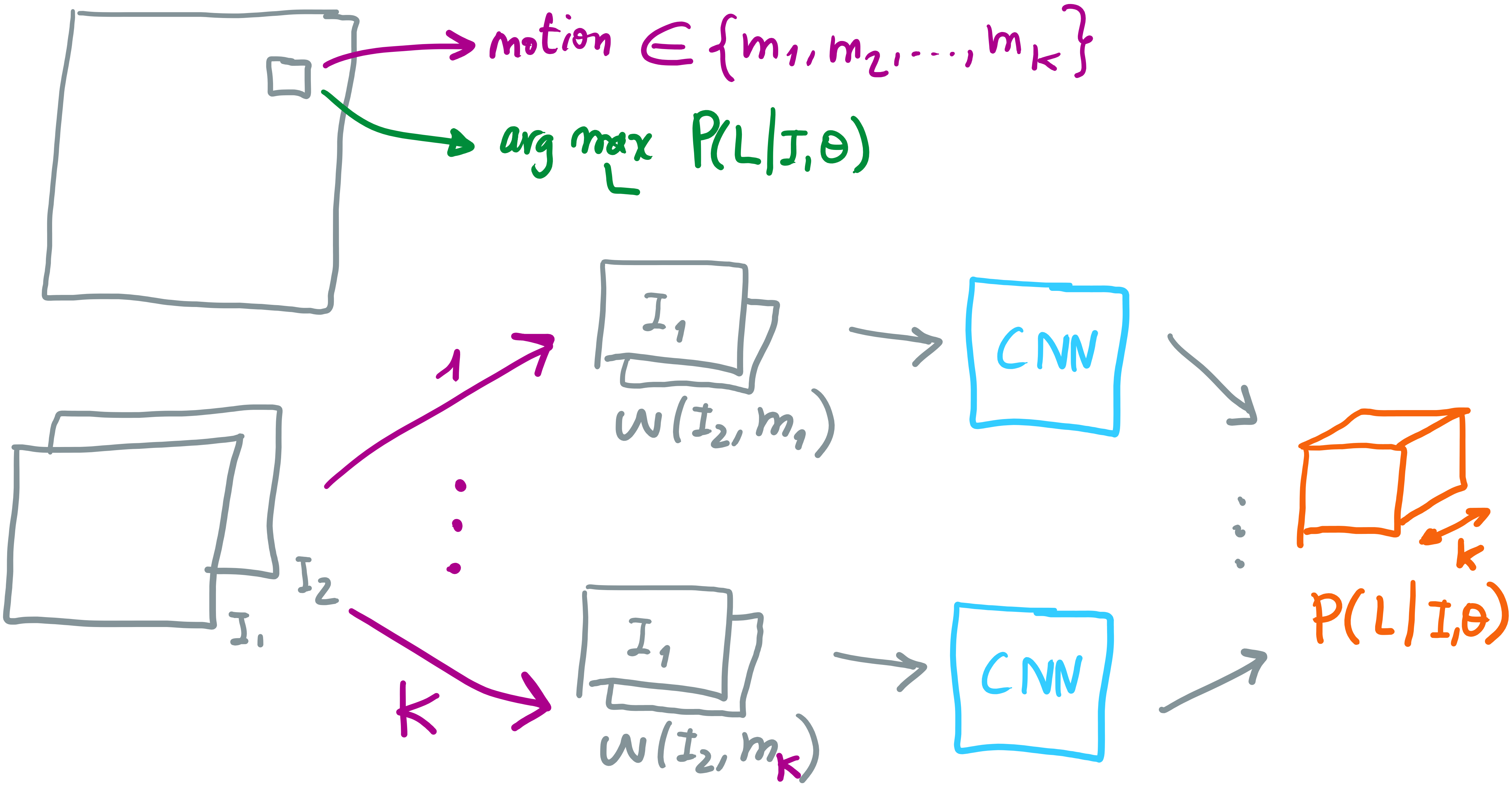
- Each G_i trained independently (previous G_i already trained)
- Since each G_i assumes small motion (simpler task), less overall parameters are required

[Optical Flow Estimation using a Spatial Pyramid Network (CVPR 2017)]

LikeNet (BMVC 2018)

- Motion as dense ***classification*** problem
- **Unsupervised** (no ground truth required!)
- **Siamese** architecture

LikeNet: A Siamese Motion Estimation Network Trained in an Unsupervised Way (BMVC 2018)



motion $\in \{m_1, m_2, \dots, m_K\}$
$\arg\max_L P(L|I,\theta)$
1
K
I_1
I_2
I_1
$w(I_2, m_1)$
I_1
$w(I_2, m_K)$
CNN
CNN
K
$P(L|I,\theta)$

Loss function?

We don't have ground truth to compare with!

$$C(I;\theta) = \sum_{i} \sum_{\mathbf{m}_k \in M} P(L_i = \mathbf{m}_k | I, \theta) D(i, \mathbf{m}_k)$$

$$D(i, \mathbf{m}_k) = JSD(\mathbf{F}(x_i + \mathbf{m}_k, t + dt) \| \mathbf{F}(x_i, t))$$

D(·) could be a function of image **values**,
but with **features** works better. Any idea **why**?

Motion should be quantised!

How many branches would be required for 50 values in t_x and t_y each?

Multiscale approach: # branches: 121, 169, 49, 9, 9

More branches at lowest resolution

Comparison

Method	Number of learned parameters
FlowNetS	32,070,472
FlowNetC	32,561,032
SpyNet	1,200,250
LikeNet	**697,028**

- performs **better** than the *other unsupervised* methods
- **generalizes** well to unknown datasets (without finetunning)