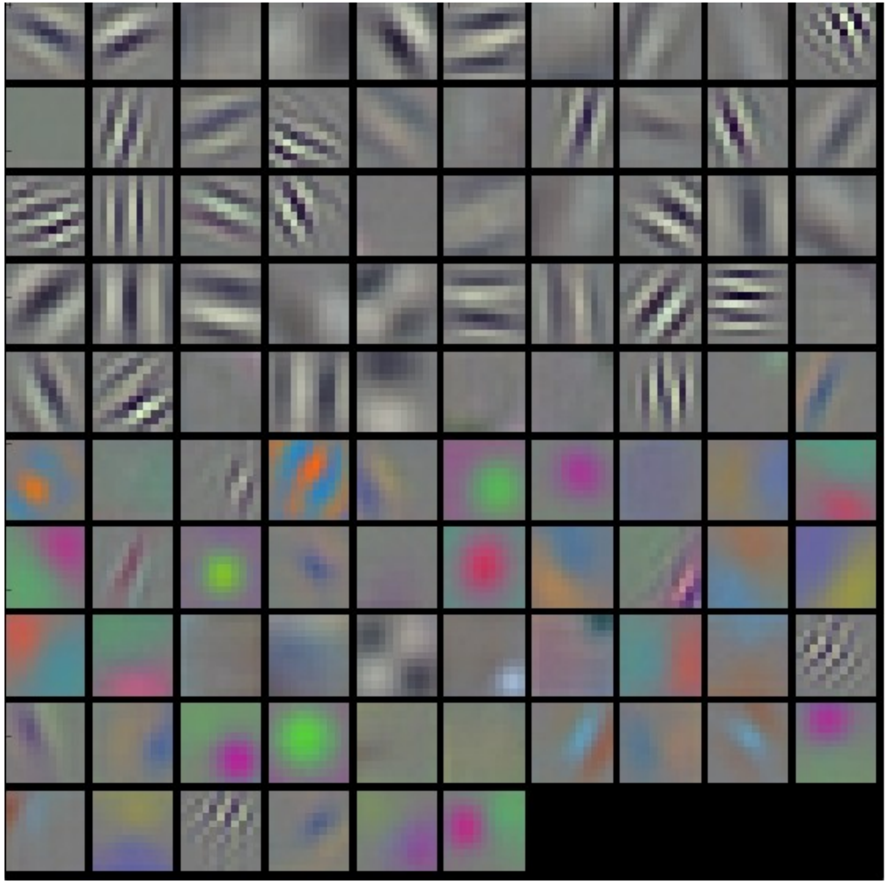
Visualization of ConvNets

Visualization of ConvNets

- Visualization of Features
- Visualization of Activations
- Visualization of Gradients
- T-SNE Visualization
- DeepDream
- ….

Visualization is a great way for debugging!

Visualization of Features



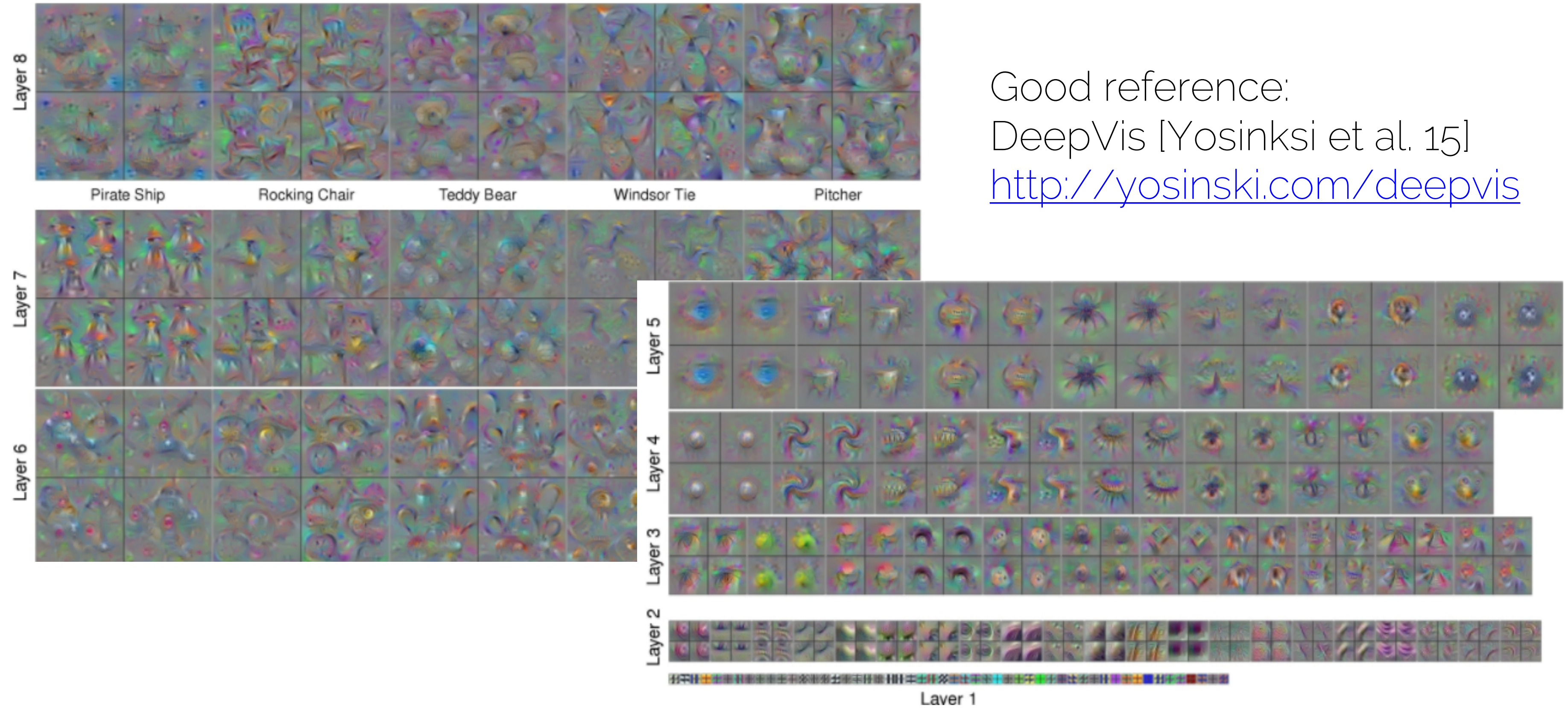
Visualization of AlexNet Features
first Conv Layer (weights visualized)

Color clusters are due to AlexNet streams

Other layers are not so easy to visualize typically need projection first

Credit: Li/Karpathy/Johnson

Visualization of Gradients



Good reference:
DeepVis [Yosinksi et al. 15]
http://yosinski.com/deepvis

Deep Visualization Toolbox

yosinski.com/deepvis

#deepvis

Jason Yosinski

Jeff Clune

Anh Nguyen

Thomas Fuchs

Hod Lipson

Cornell University

University of Wyoming

Jet Propulsion Laboratory
California Institute of Technology

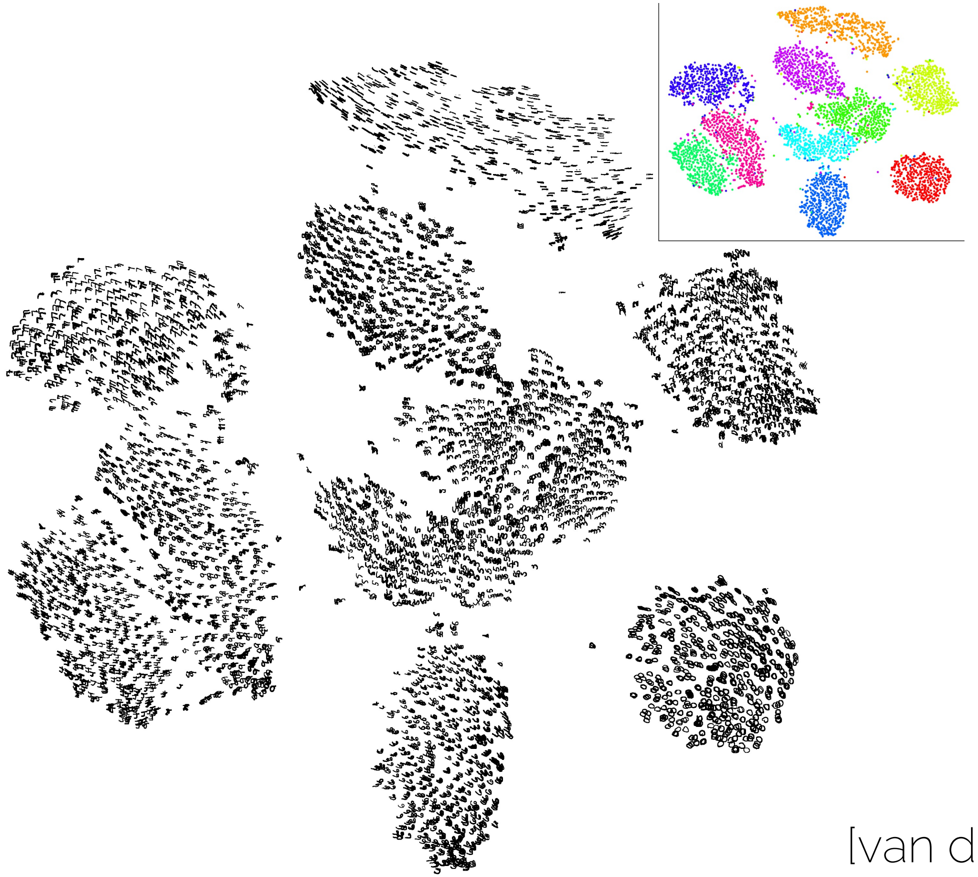
t-SNE Visualization

t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Map high-dimensional embedding to 2D map
- Add samples from dataset according to their features to large image
- Very useful to spot clusters and debug embedding

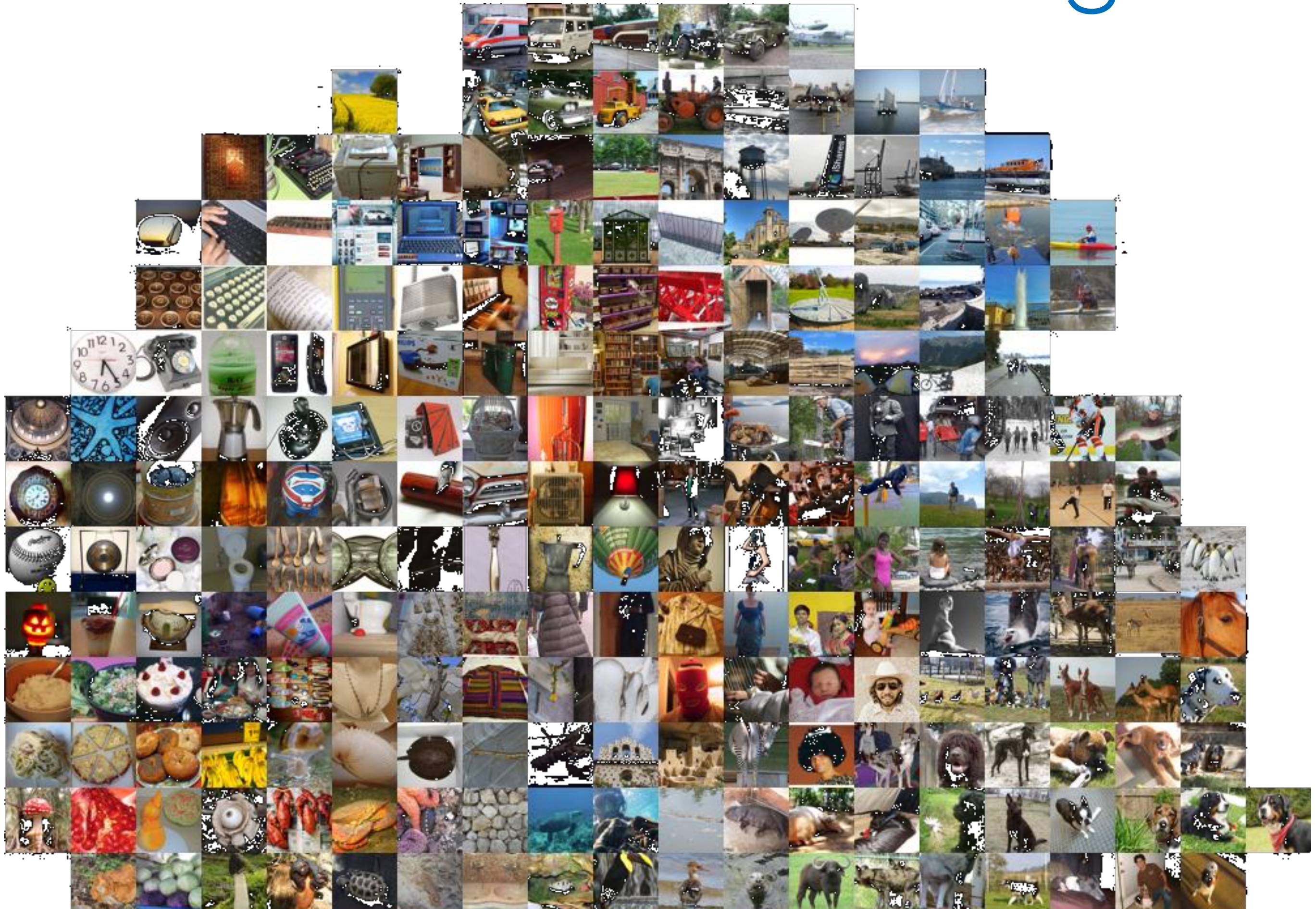
[van der Maaten et al.] t-SNE

t-SNE Visualization: MNIST



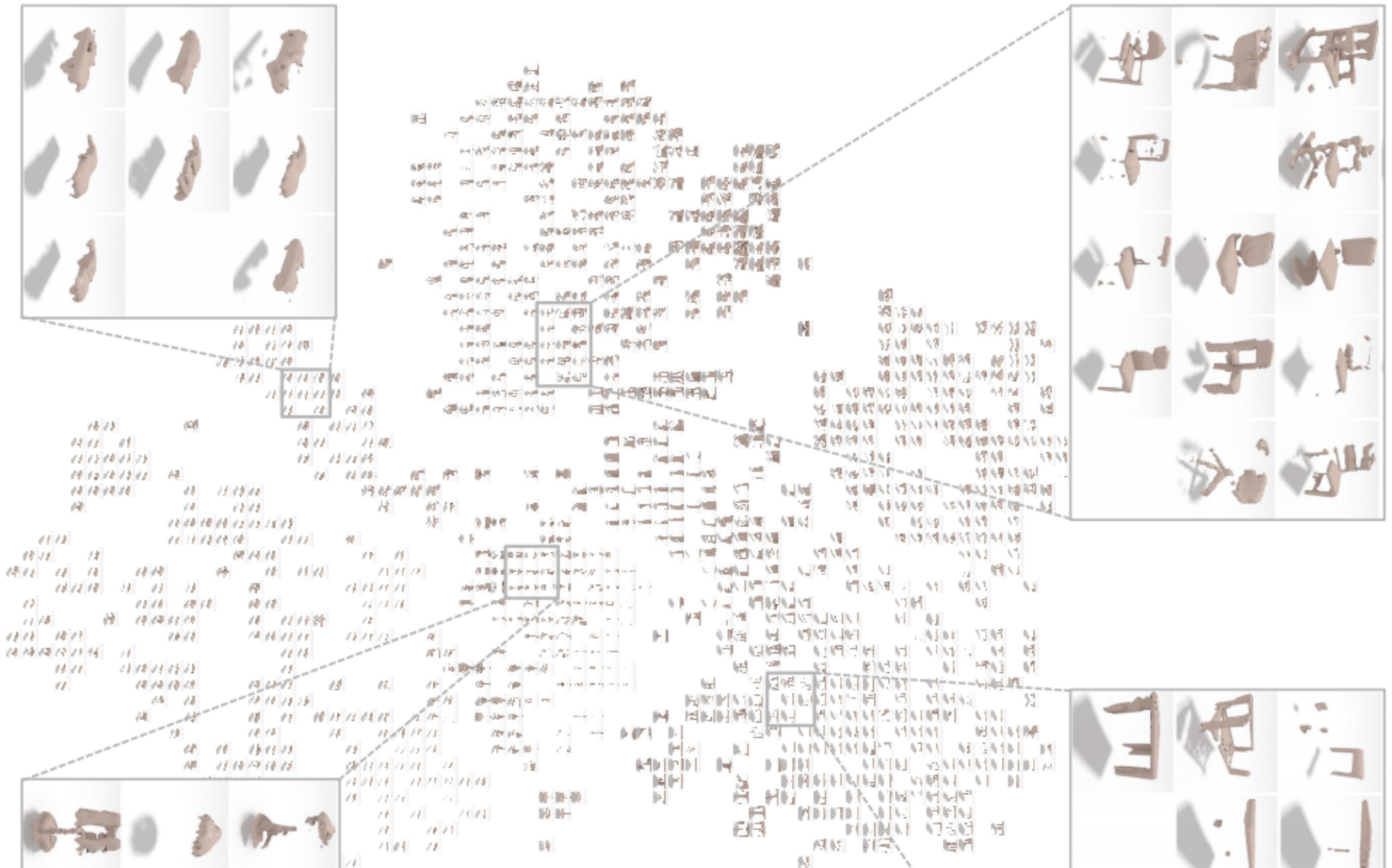
[van der Maaten et al.] t-SNE

t-SNE Visualization: ImageNet



Karpathy

t-SNE Visualization: ShapeNet



DeepDream

[Mordvintsev et al. 15] DeepDream

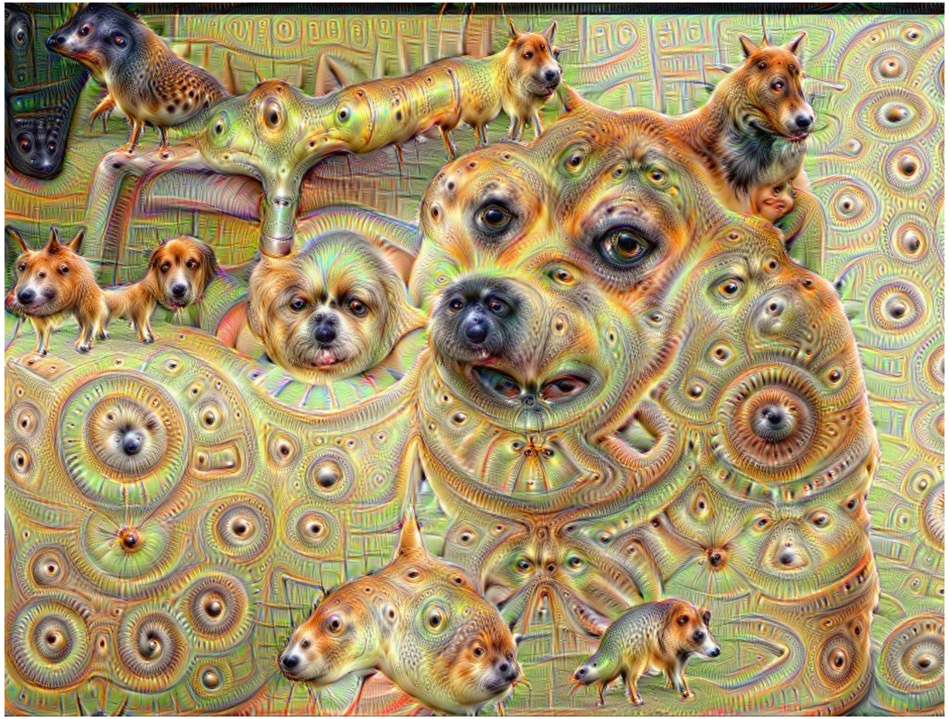
DeepDream

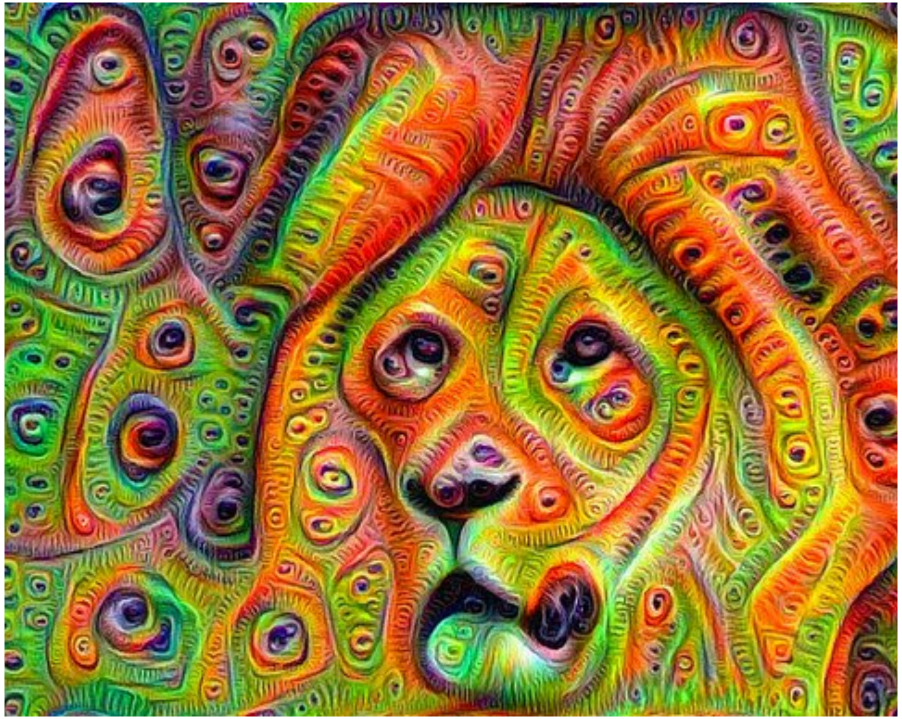
- Forward pass to the layer where you want to dream
- Make the gradients = raw activations
- Backprop to the image

- Amplifying the features that were activated at that layer
- Repeat

[Mordvintsev et al. 15] DeepDream

DeepDream

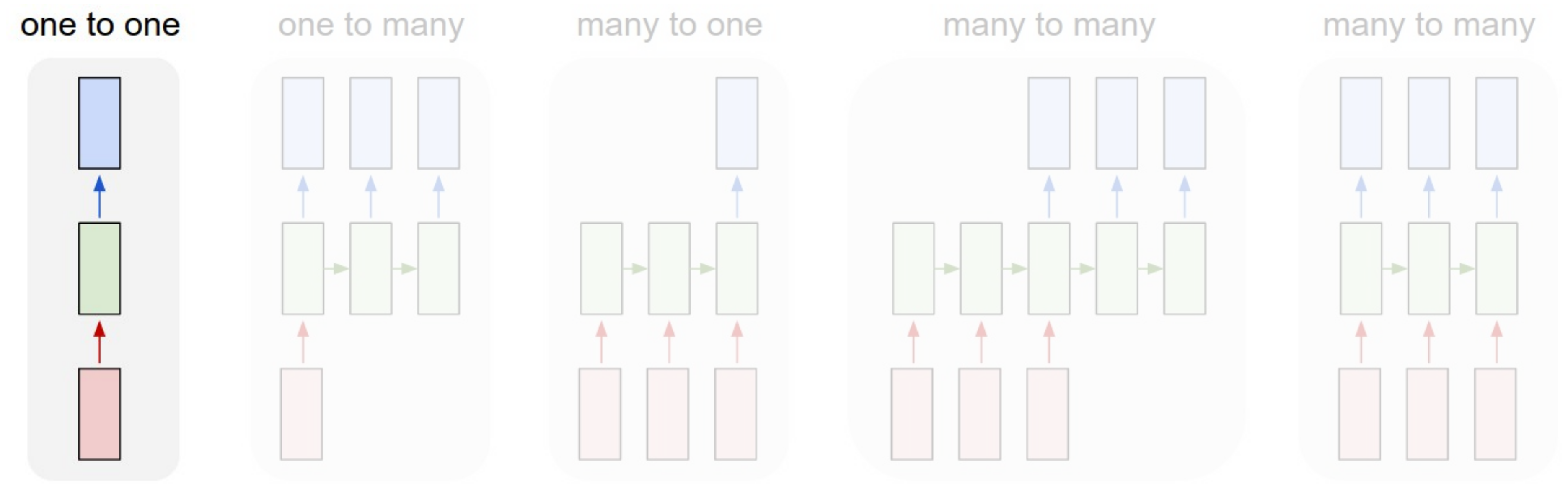




[Mordvintsev et al. 15] DeepDream

Recurrent Neural Networks

RNNs are flexible



Classic Neural Networks for Image Classification

RNNs are flexible

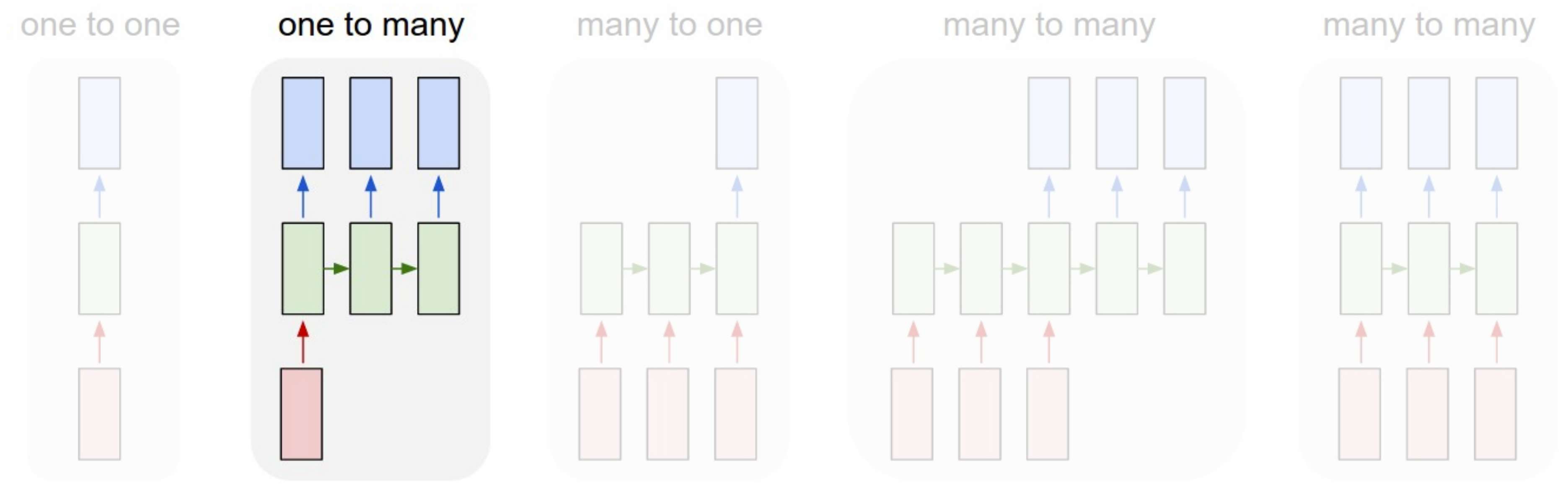
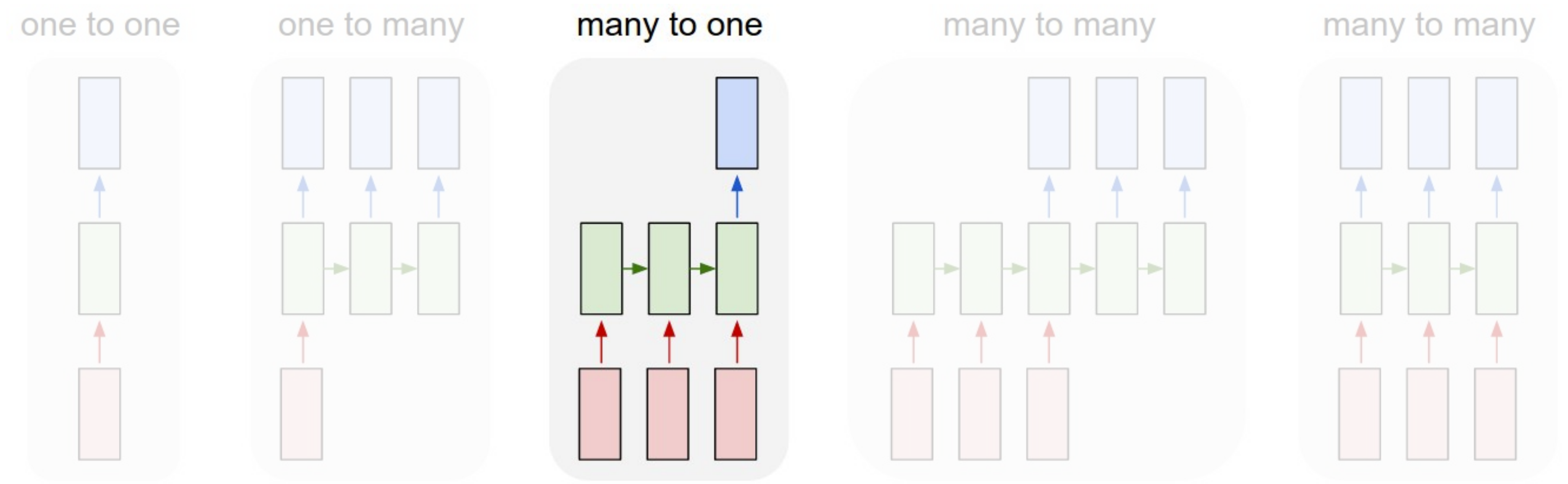


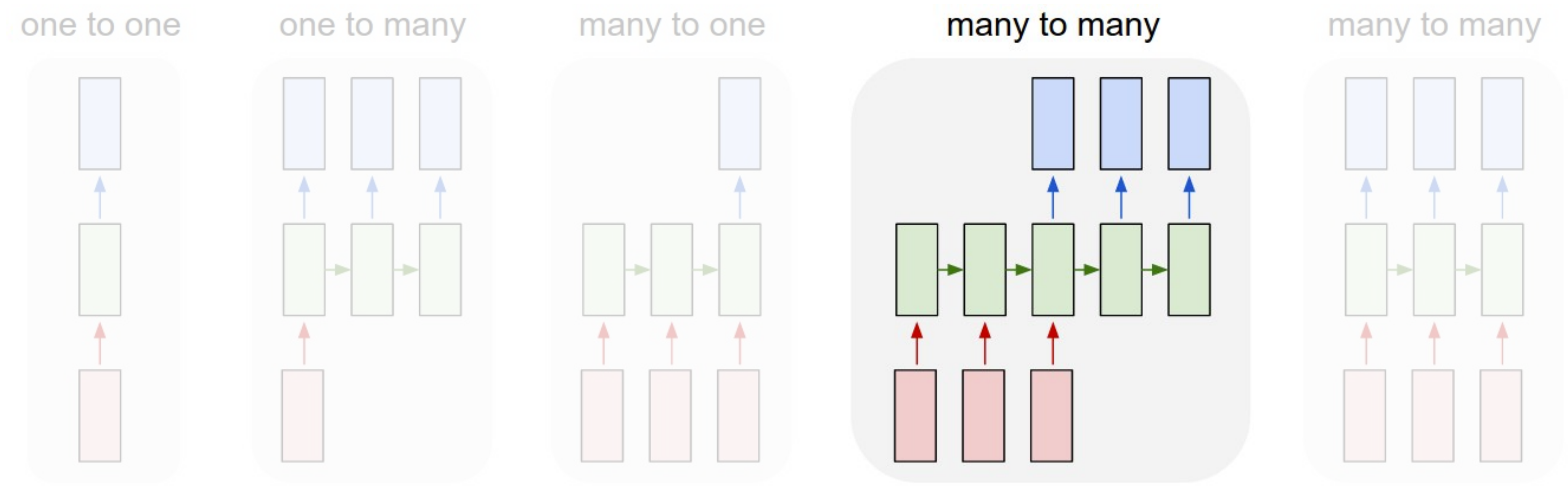
Image captioning

RNNs are flexible



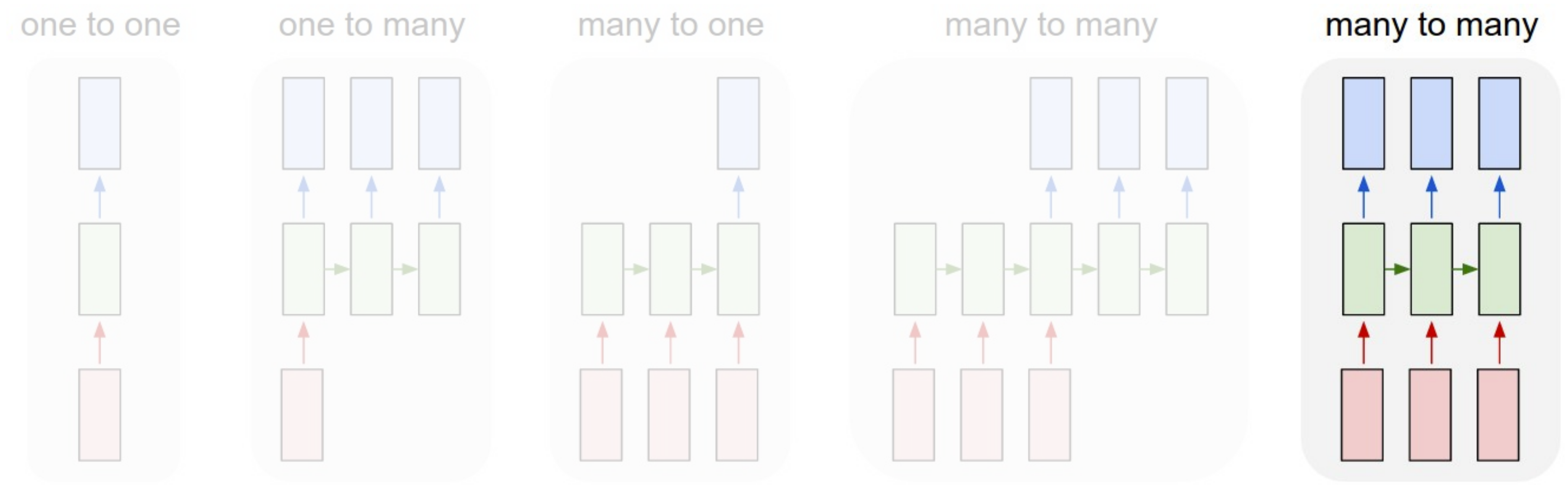
Language recognition

RNNs are flexible



Machine translation

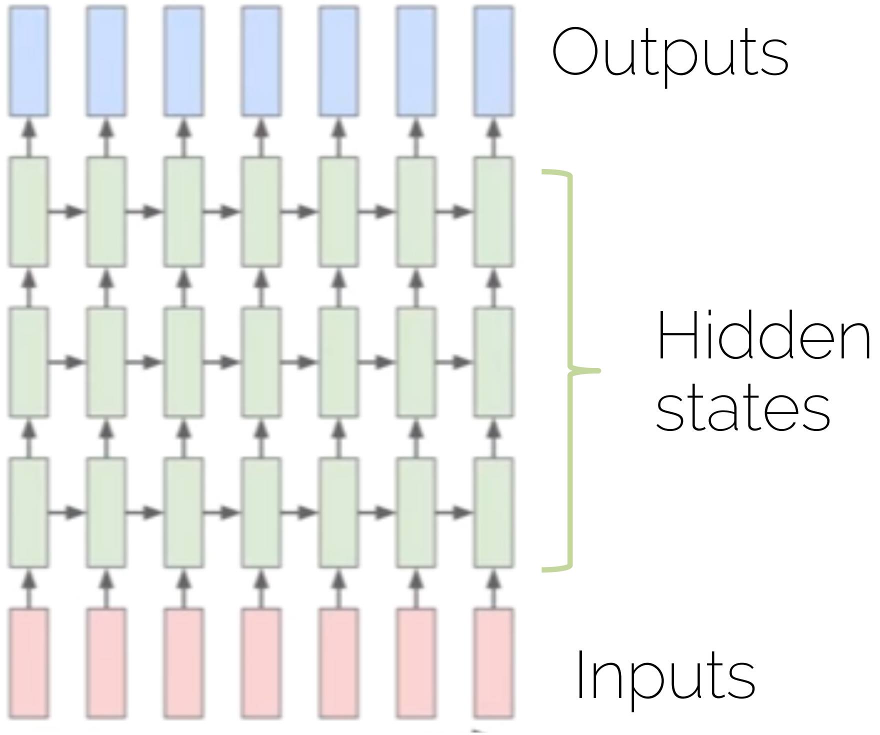
RNNs are flexible



Event classification

Basic structure of a RNN

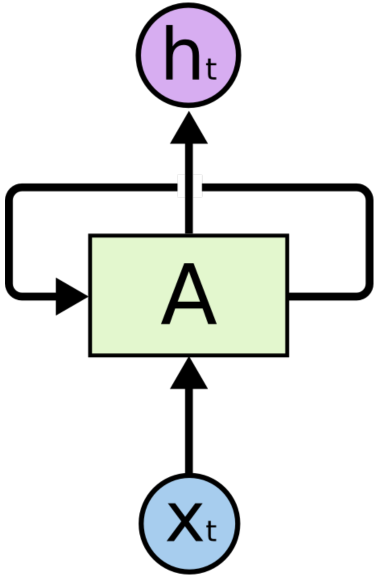
- Multi-layer RNN



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

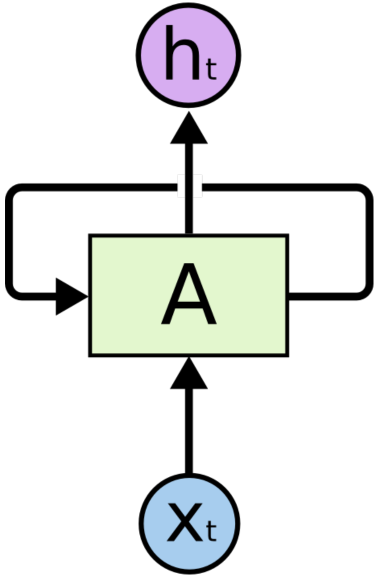
input

[Christopher Olah] Understanding LSTMs

Basic structure of a RNN

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

Hidden state



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

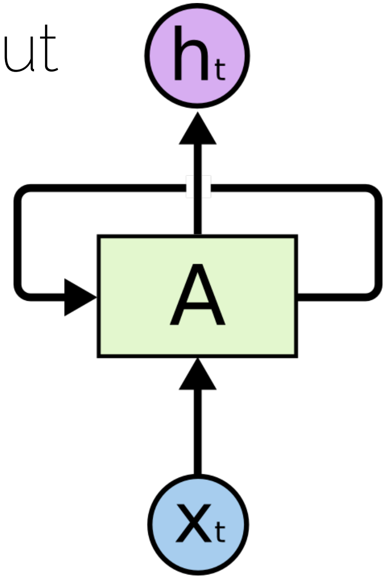
Parameters to be learned

Basic structure of a RNN

- We want to have notion of “time” or “sequence”

Output

Hidden state



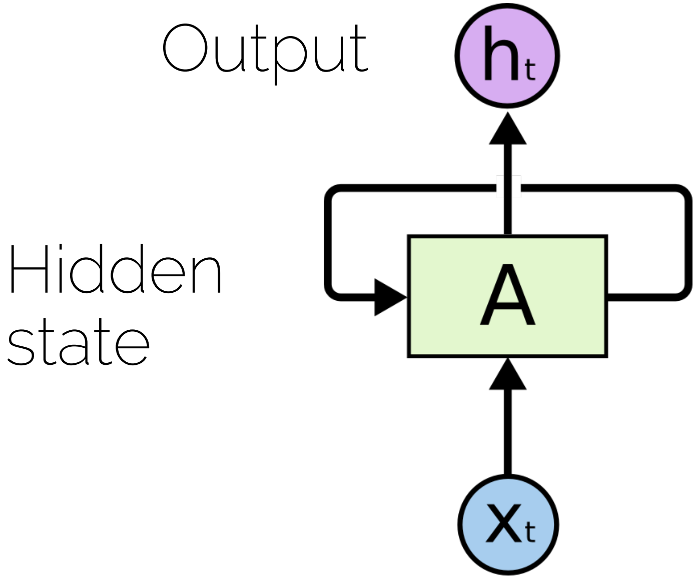
$$\mathbf{A}_t = \boldsymbol{\theta}_c \mathbf{A}_{t-1} + \boldsymbol{\theta}_x \mathbf{x}_t$$

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

Note: non-linearities ignored for now

Basic structure of a RNN

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



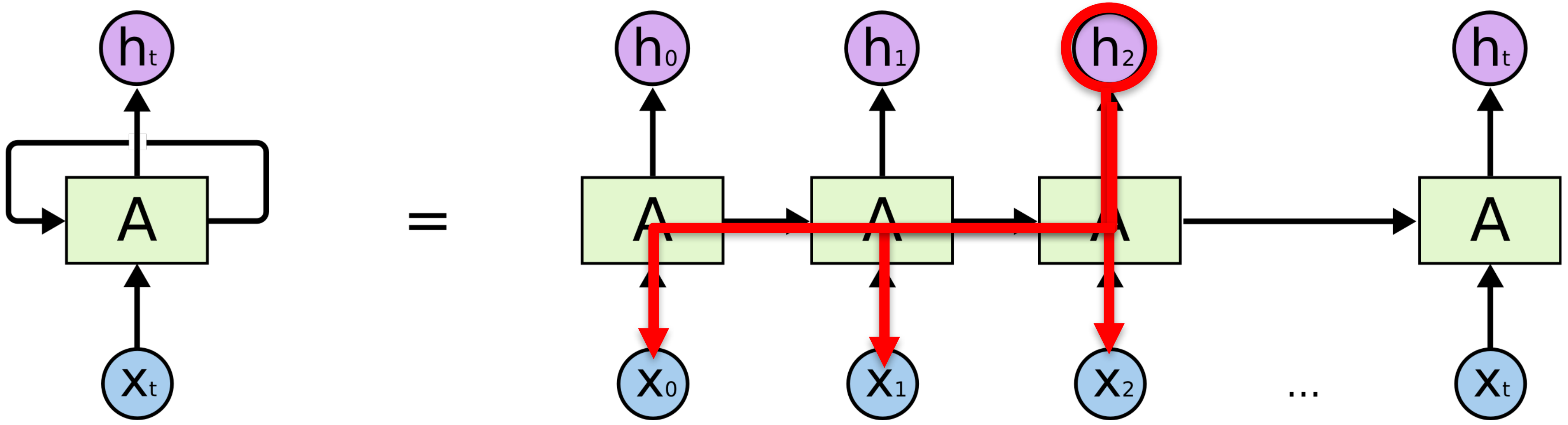
$$\mathbf{A}_t = \boldsymbol{\theta}_c \mathbf{A}_{t-1} + \boldsymbol{\theta}_x \mathbf{x}_t$$

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

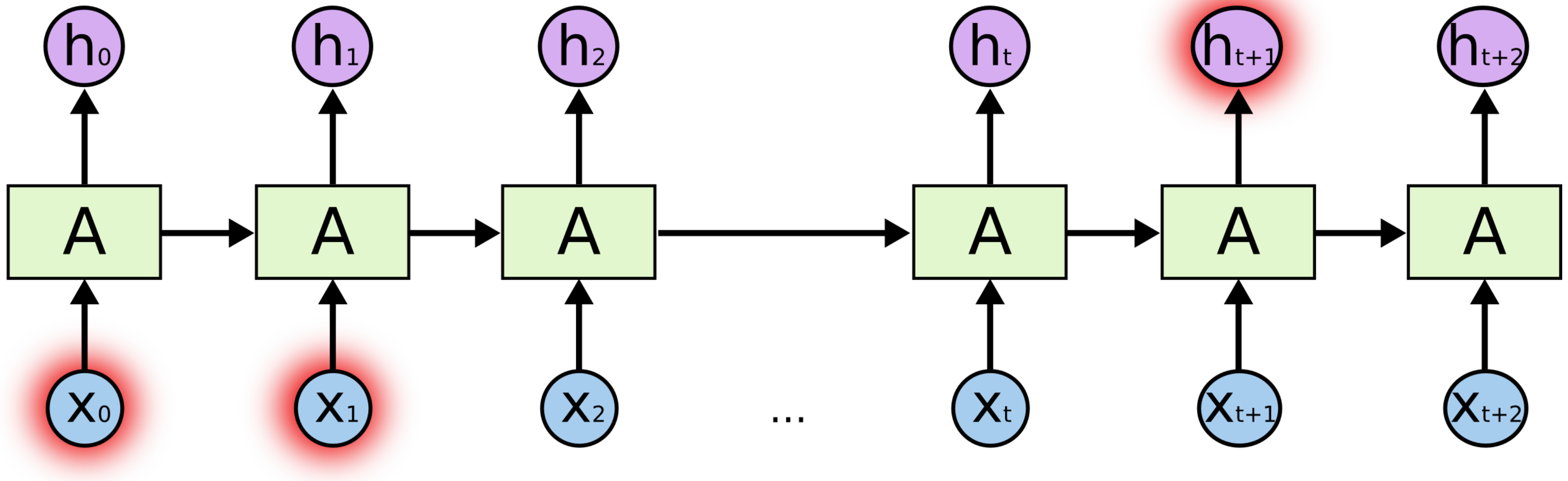
Same parameters for each time step = generalization!

Basic structure of a RNN

- Unrolling RNNs



Long-term dependencies

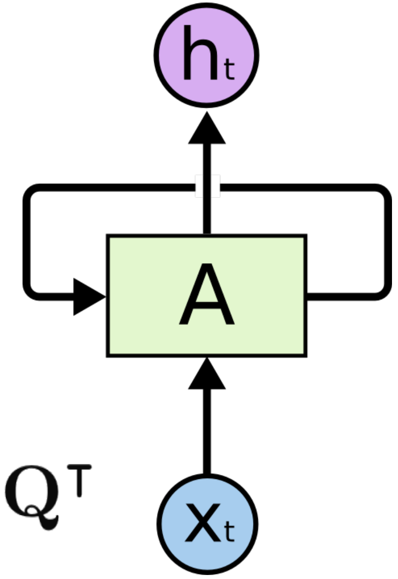


I moved to Germany … so I speak German fluently

Long-term dependencies

- Simple recurrence $\mathbf{A}_t = \boldsymbol{\theta}_c \mathbf{A}_{t-1} + \boldsymbol{\theta}_x \mathbf{x}_t$
- Let us forget the input $\mathbf{A}_t = \boldsymbol{\theta}^t \mathbf{A}_0$
- If $\boldsymbol{\theta}$ admits eigendecomposition $\boldsymbol{\theta} = \mathbf{Q}\Lambda\mathbf{Q}^\intercal$
- Orthogonal $\boldsymbol{\theta}$ allows us to simplify the recurrence

$$\mathbf{A}_t = \mathbf{Q}\Lambda^t\mathbf{Q}^\intercal \mathbf{A}_0$$



Long-term dependencies

- Simple recurrence $\mathbf{A}_t = \mathbf{Q}\Lambda^t\mathbf{Q}^\intercal\mathbf{A}_0$

What happens to eigenvalues with magnitude less than one?

Vanishing gradient

What happens to eigenvalues with magnitude larger than one?

Exploding gradient

A simple solution

- Let us just make it 1 $\mathbf{A}_t = \boldsymbol{\theta}_c \mathbf{A}_{t-1} + \boldsymbol{\theta}_x \mathbf{x}_t$

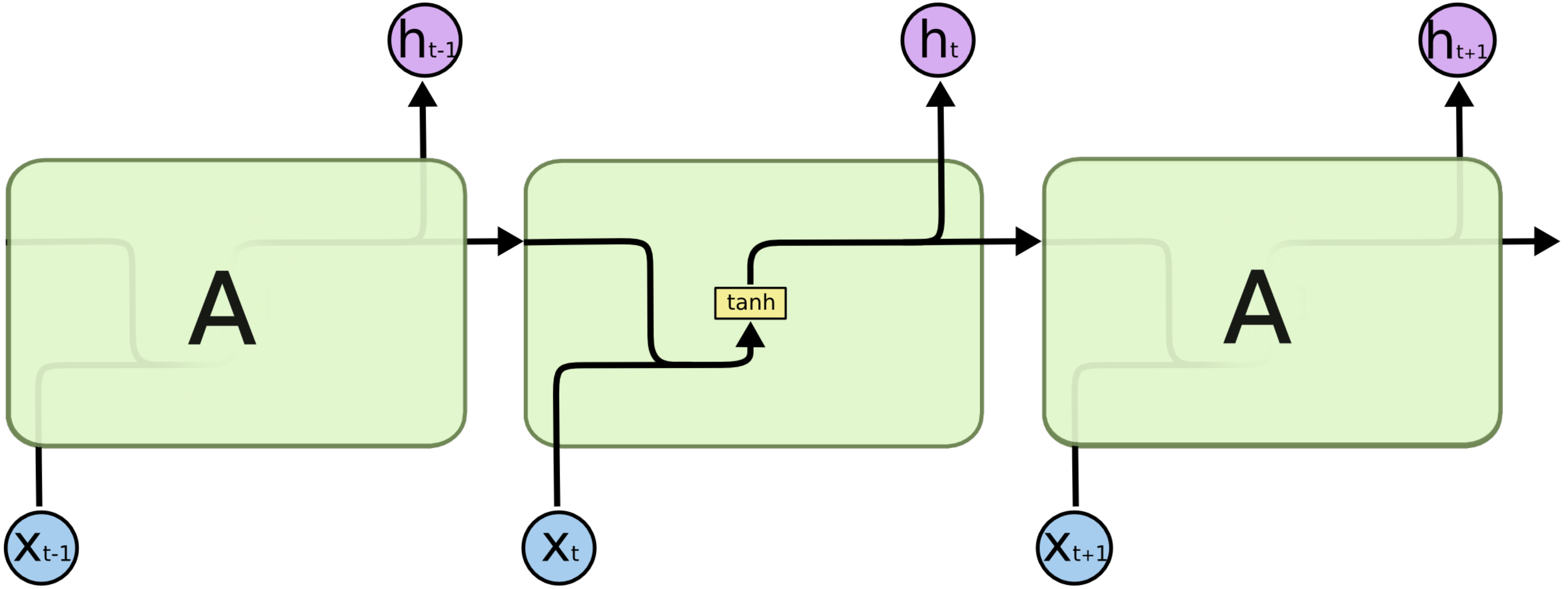
- The previous step will always has a “positive” impact on the current step. What if we want to forget some information?

Long Short Term Memory

Hochreiter and Schmidhuber 1997

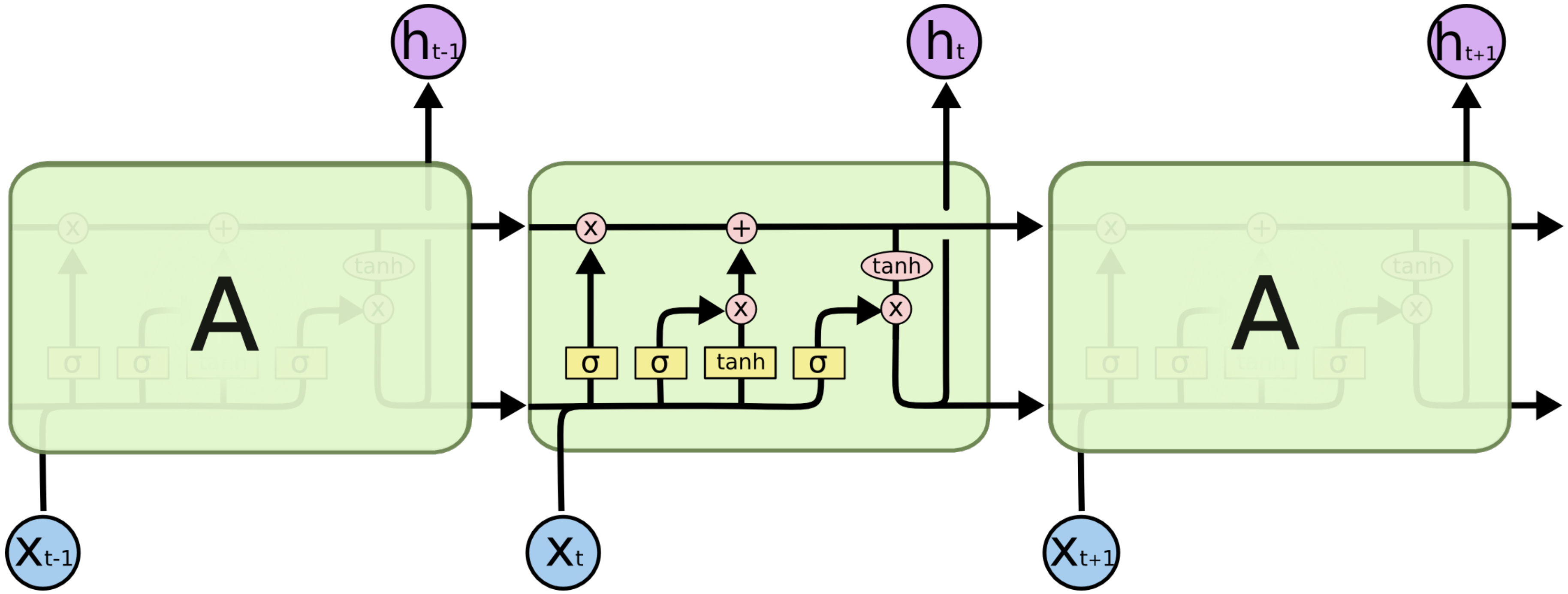
Long-Short Term Memory Units

- Simple RNN has tanh as non-linearity



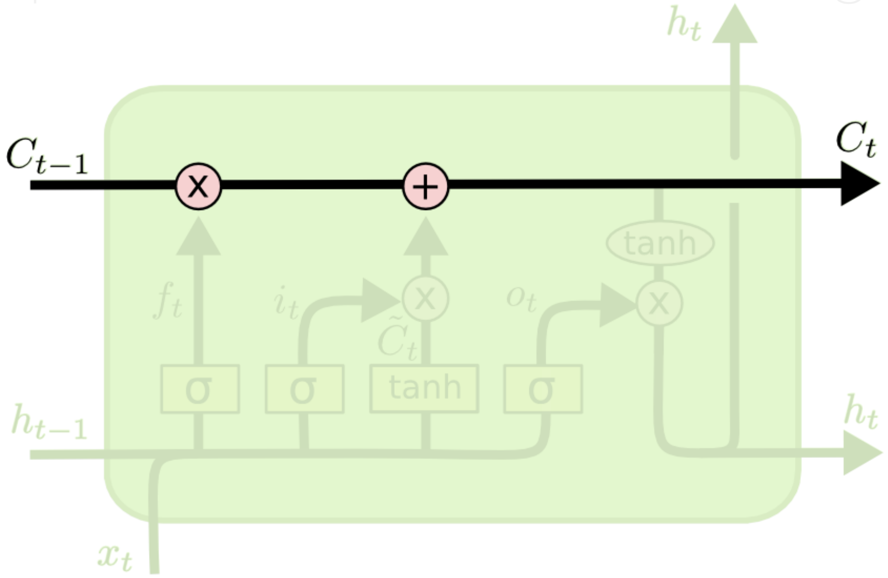
Long-Short Term Memory Units

- LSTM



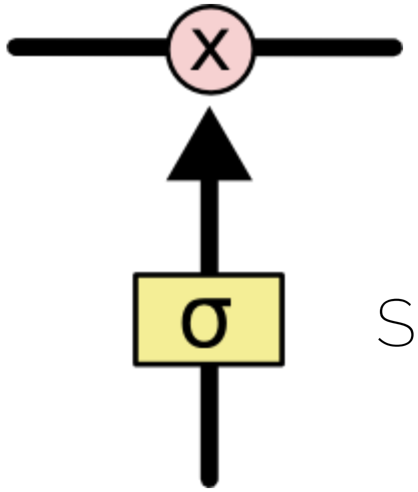
Long-Short Term Memory Units

- Key ingredients
- Cell = transports the information through the unit



Long-Short Term Memory Units

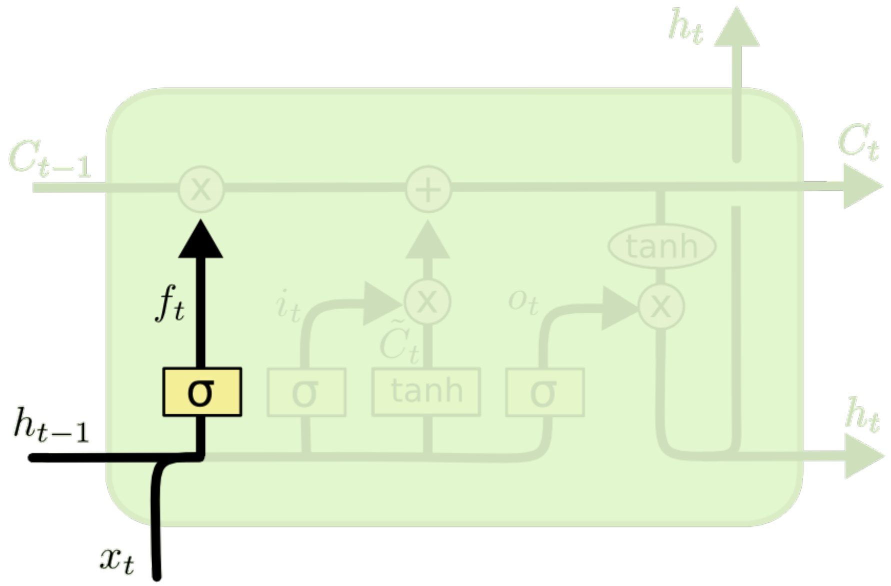
- Key ingredients
- Cell = transports the information through the unit
- Gate = remove or add information to the cell state



Sigmoid

LSTM: step by step

- Forget gate

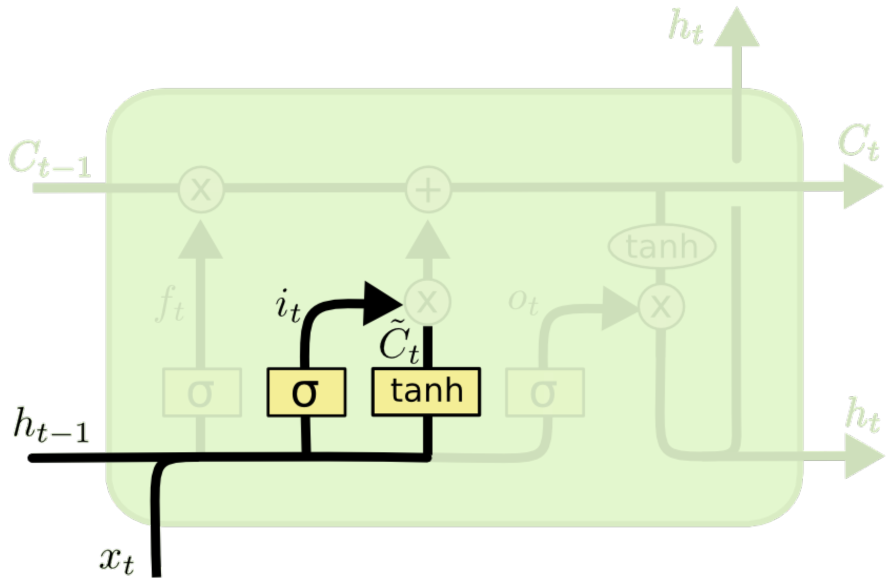


Decides when to erase the cell state

Sigmoid = output between 0 (forget) and 1 (keep)

LSTM: step by step

- Input gate

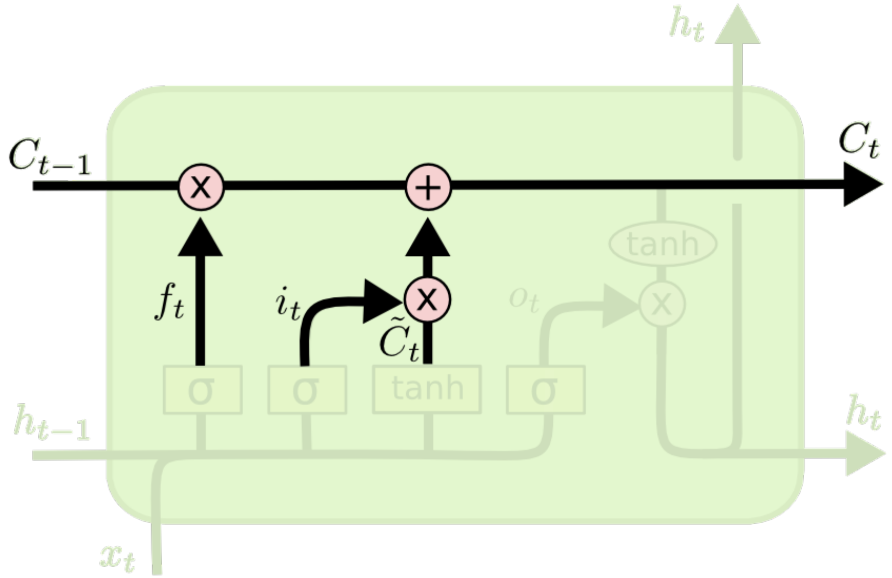


Decides which values will be updated

New cell state, output from a tanh (-1,1)

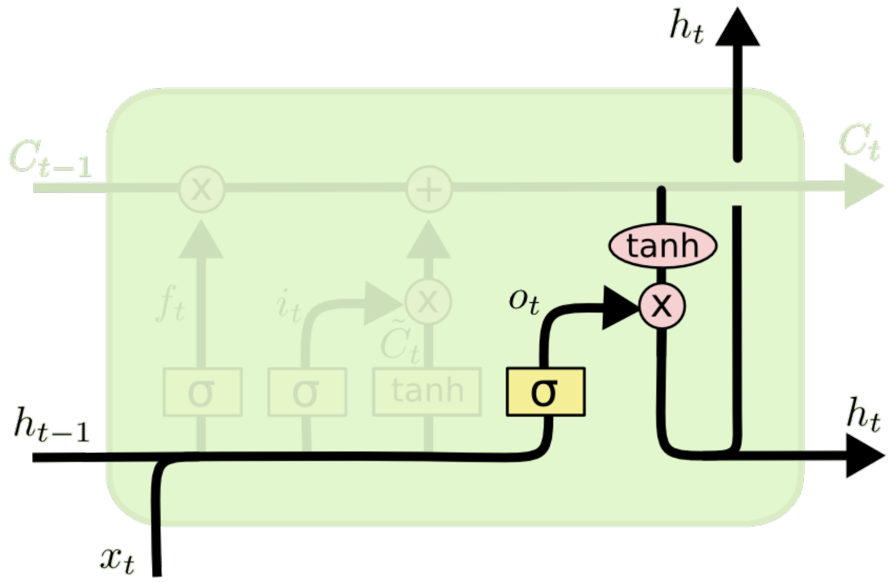
LSTM: step by step

- Element-wise operations



LSTM: step by step

- Output gate



Decides which values will be outputted

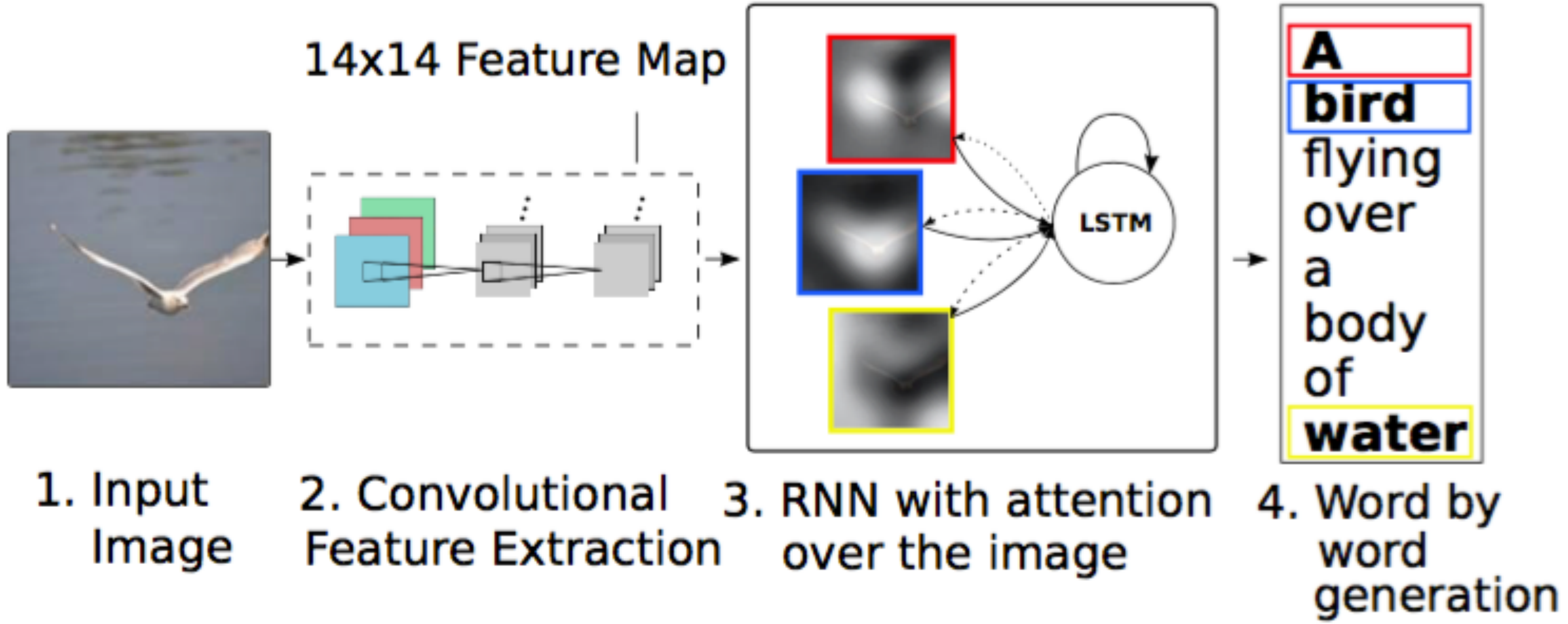
Output from a tanh (-1,1)

LSTM: step by step

- Forget gate $\mathbf{f}_t = Sigm(\boldsymbol{\theta}_{xf}\mathbf{x}_t + \boldsymbol{\theta}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f)$
- Input gate $\mathbf{i}_t = Sigm(\boldsymbol{\theta}_{xi}\mathbf{x}_t + \boldsymbol{\theta}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i)$
- Output gate $\mathbf{o}_t = Sigm(\boldsymbol{\theta}_{xo}\mathbf{x}_t + \boldsymbol{\theta}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o)$
- Cell update $\mathbf{g}_t = Tanh(\boldsymbol{\theta}_{xg}\mathbf{x}_t + \boldsymbol{\theta}_{hg}\mathbf{h}_{t-1} + \mathbf{b}_g)$
- Cell $\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$
- Output $\mathbf{h}_t = \mathbf{o}_t \odot Tanh(\mathbf{C}_t)$

RNN's in Computer Vision

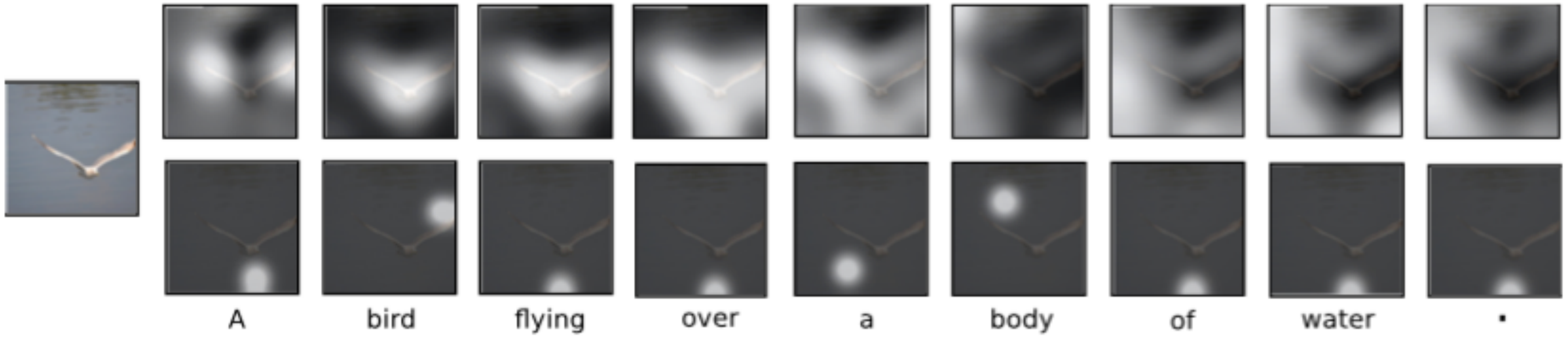
- Caption generation


14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

Xu et al. 2015

RNN's in Computer Vision

- Caption generation

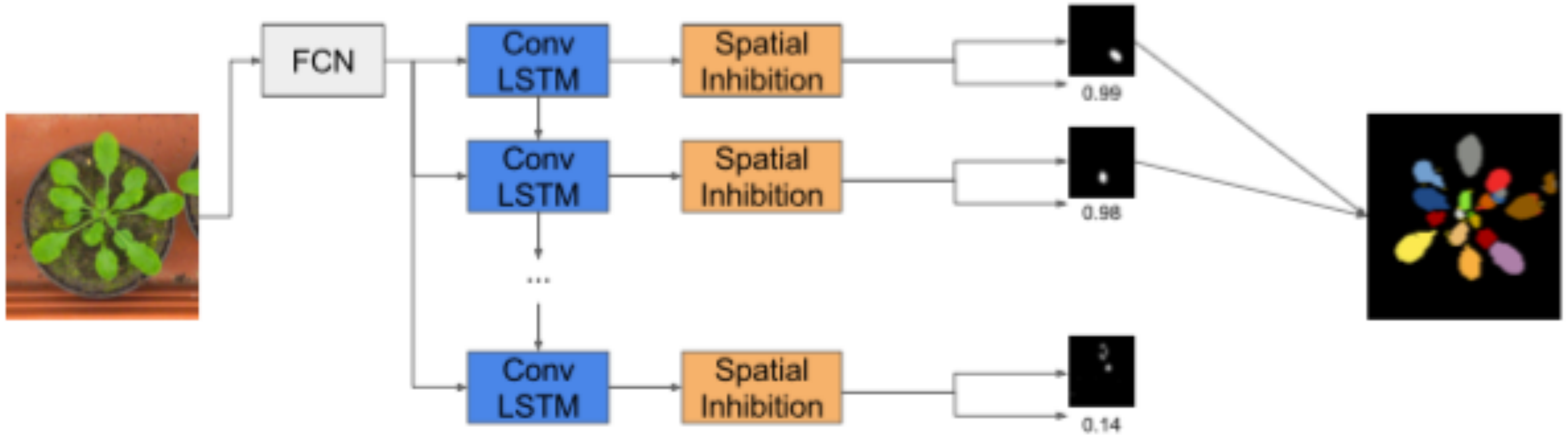


- Focus is shifted to different parts of the image

Xu et al. 2015

RNN's in Computer Vision

- Instance segmentation

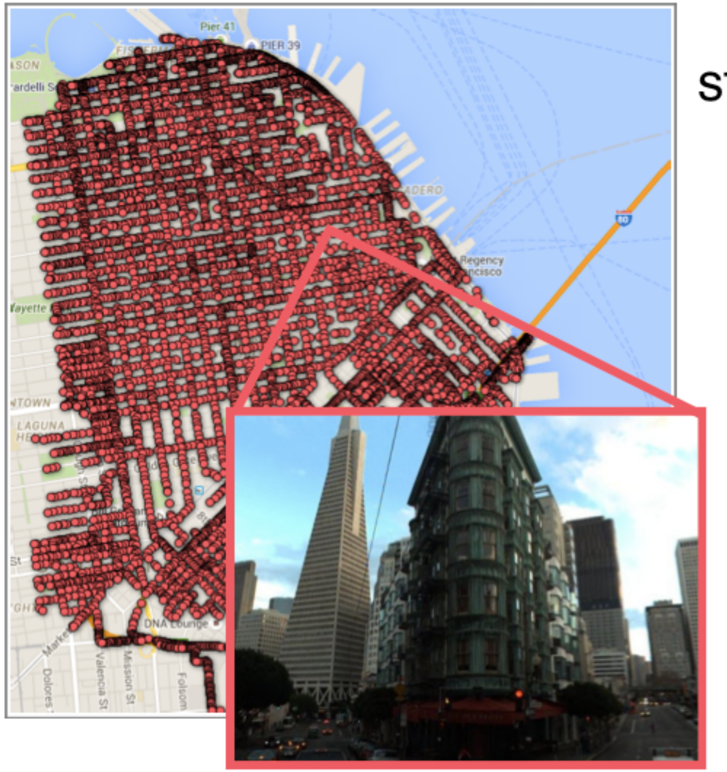


Romera-Paredes et al. 2015

RNN's in Computer Vision

- Image localization

Map



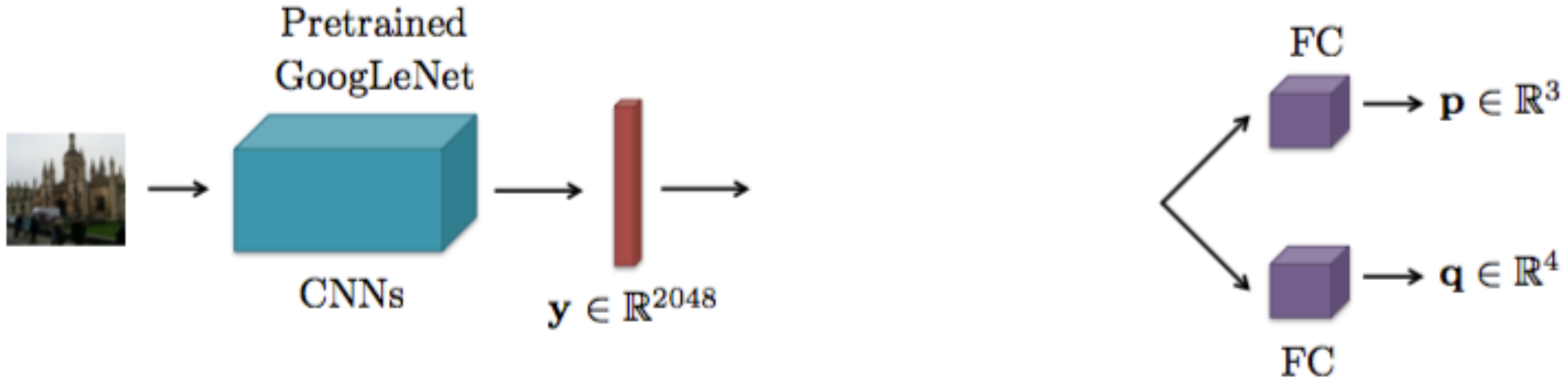
st

Photo

Walch et al 2017

RNN's in Computer Vision

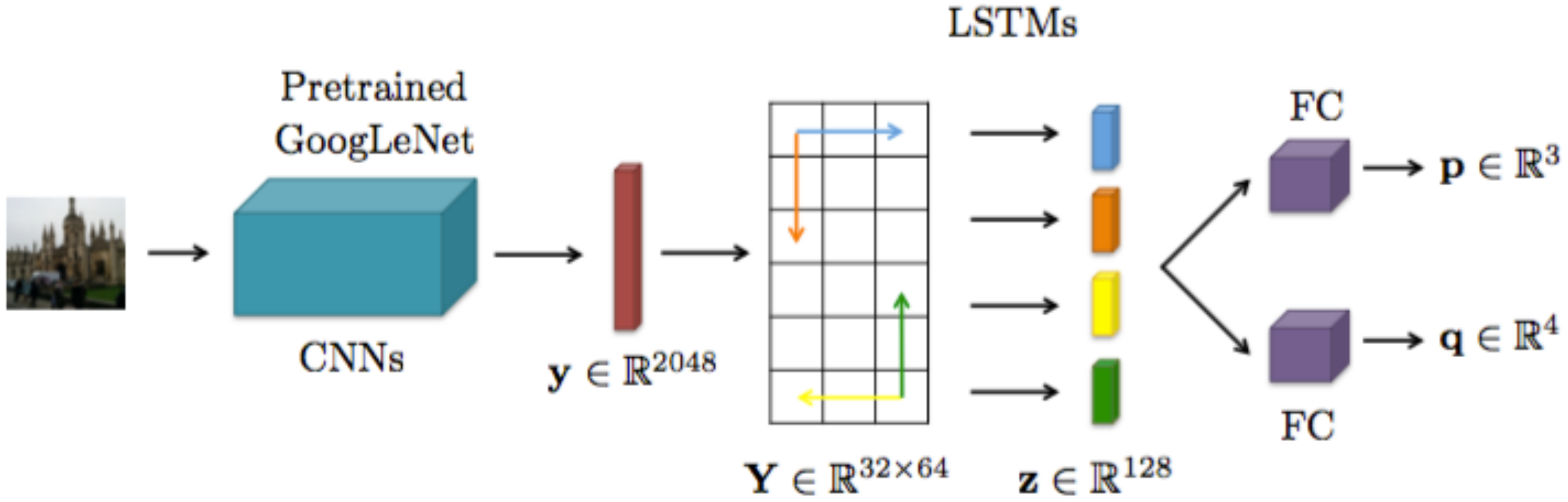
- Image localization


Pretrained GoogLeNet
CNNs
$\mathbf{y} \in \mathbb{R}^{2048}$
FC
$\mathbf{p} \in \mathbb{R}^3$
$\mathbf{q} \in \mathbb{R}^4$
FC

Walch et al 2017

RNN's in Computer Vision

- Image localization



LSTMs
Pretrained GoogLeNet
CNNs
$\mathbf{y} \in \mathbb{R}^{2048}$
$\mathbf{Y} \in \mathbb{R}^{32\times64}$
$\mathbf{z} \in \mathbb{R}^{128}$
FC
FC
$\mathbf{p} \in \mathbb{R}^3$
$\mathbf{q} \in \mathbb{R}^4$

Walch et al 2017

Administrative Things

- This was the last formal lecture!

- Next week: cool stuff we do at TUM with Deep Learning!

- In two weeks: special lecture by nVidia Autonomous Driving Team here in Munich!