# Lecture 11 - Including Attention into your Network (& Semantic Segmentation)

DD2424

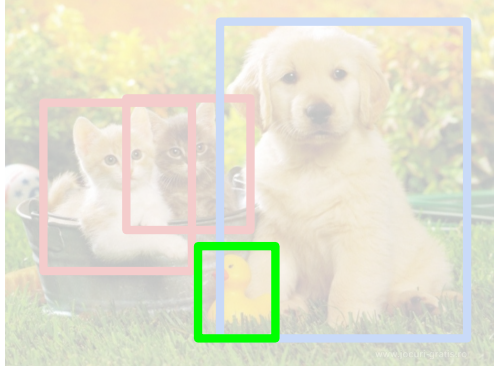
May 4, 2017

# Computer Vision Tasks

Classification

Classification + Localization

Object Detection



**Segmentation**

Today

# Semantic Segmentation

Label every pixel!

Don't differentiate instances (cows)

Classic computer vision problem

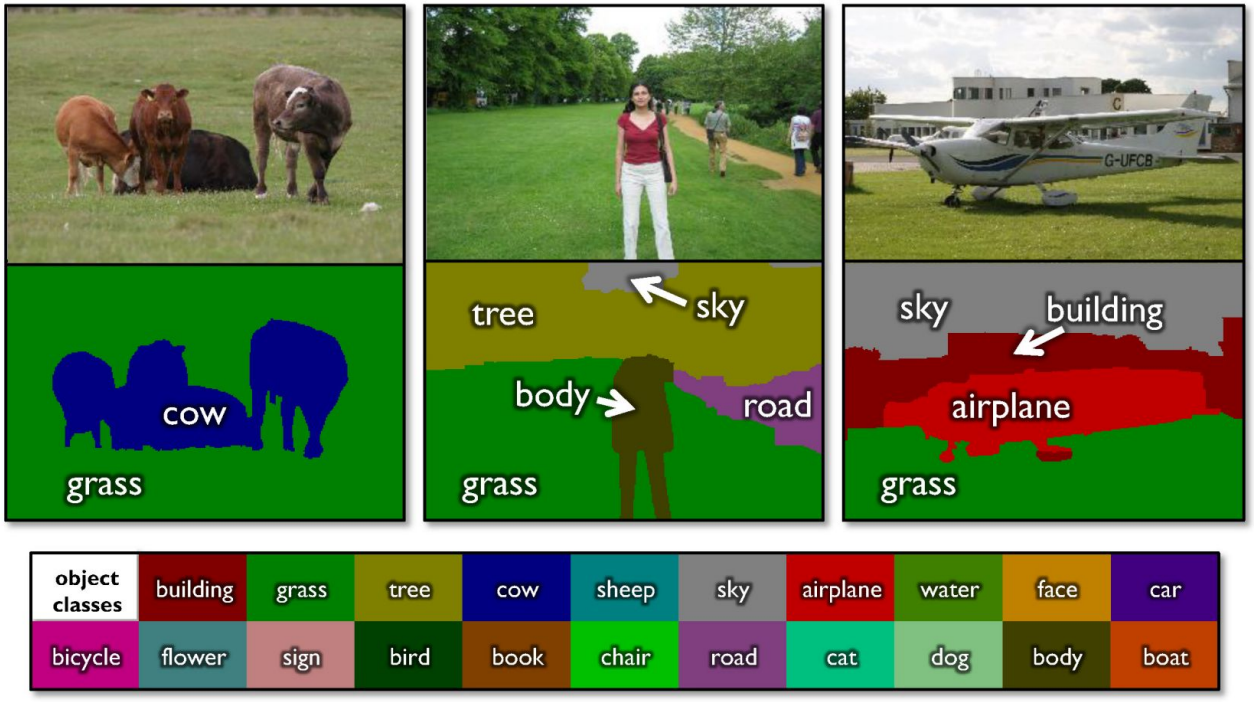



Figure credit: Shotton et al, "TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context", IJCV 2007

# Instance Segmentation

Detect instances, give category, label pixels

“simultaneous detection and segmentation” (SDS)

Lots of recent work (MS-COCO)

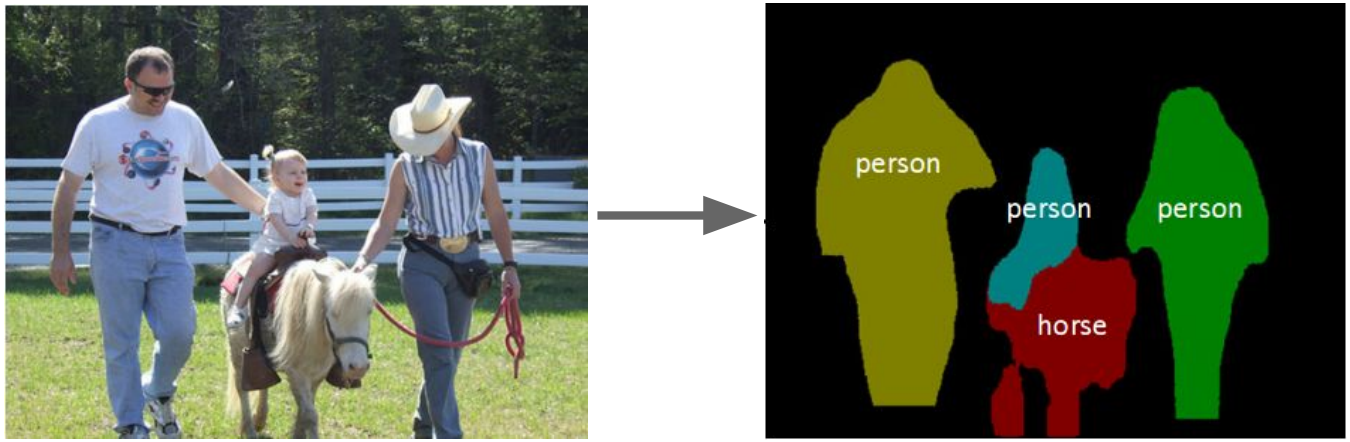


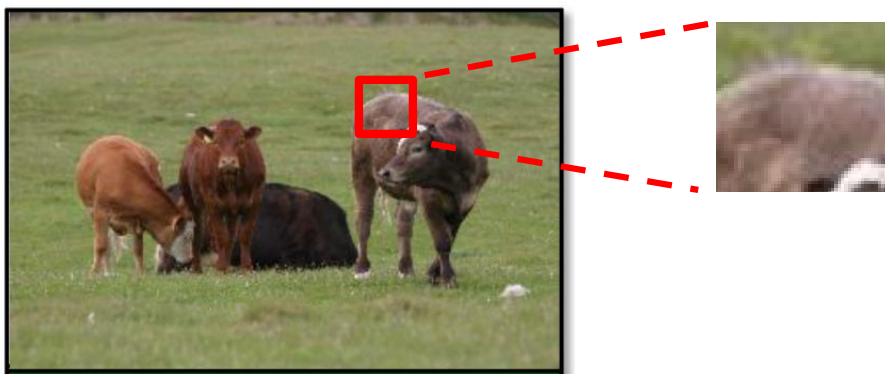

Figure credit: Dai et al, “Instance-aware Semantic Segmentation via Multi-task Network Cascades”, arXiv 2015

# Semantic Segmentation

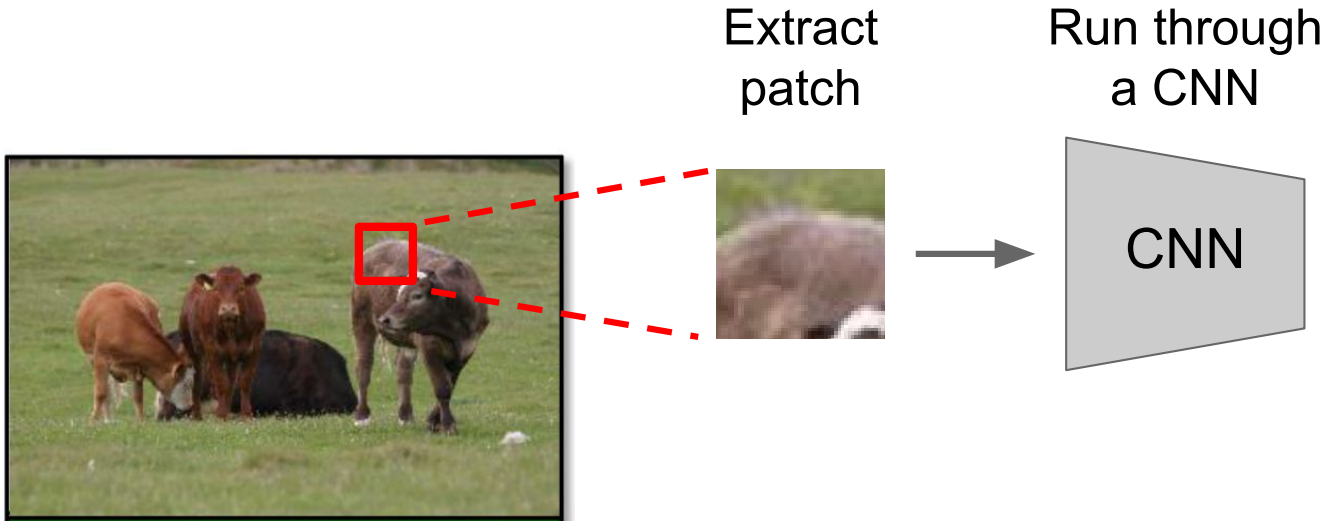
# Semantic Segmentation

# Semantic Segmentation

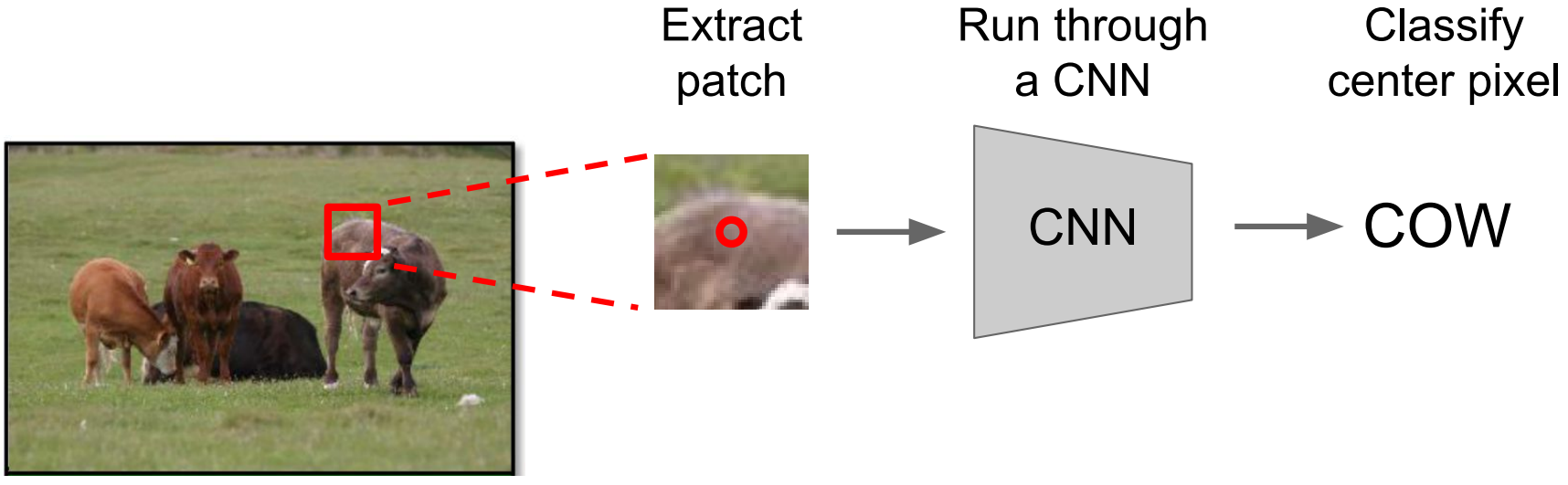
Extract patch



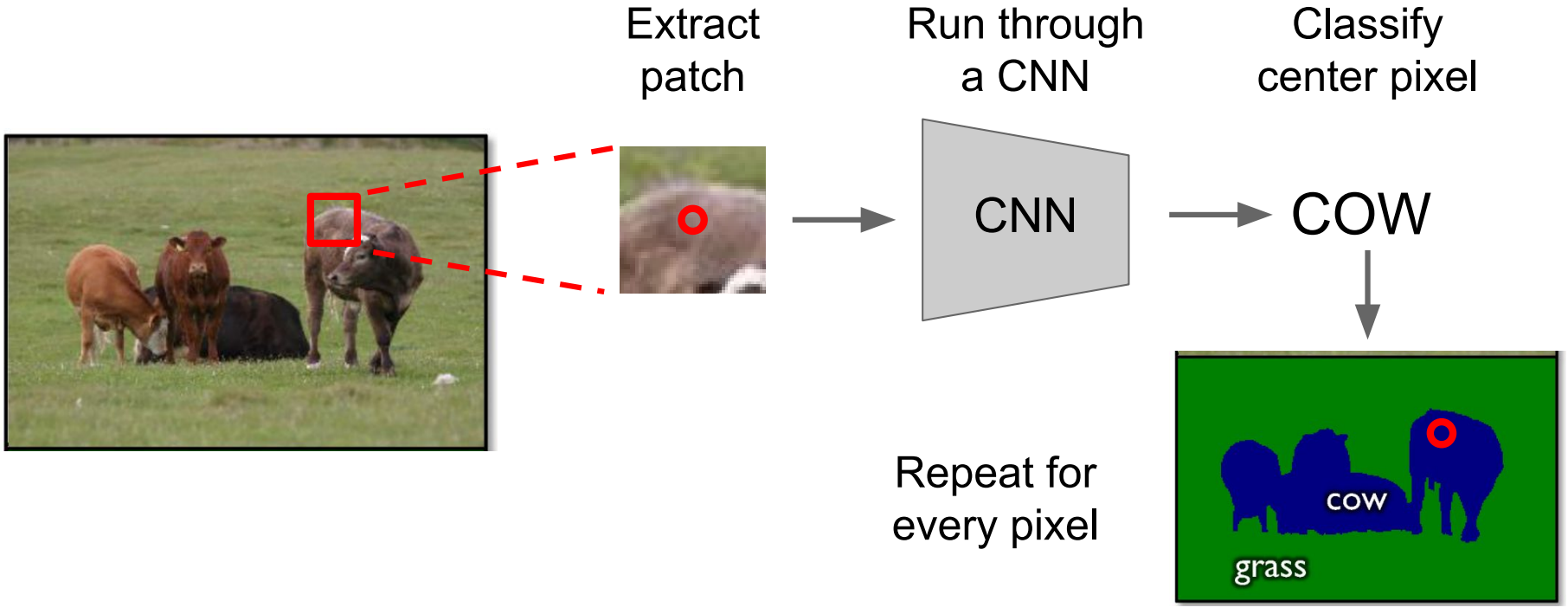
# Semantic Segmentation



# Semantic Segmentation

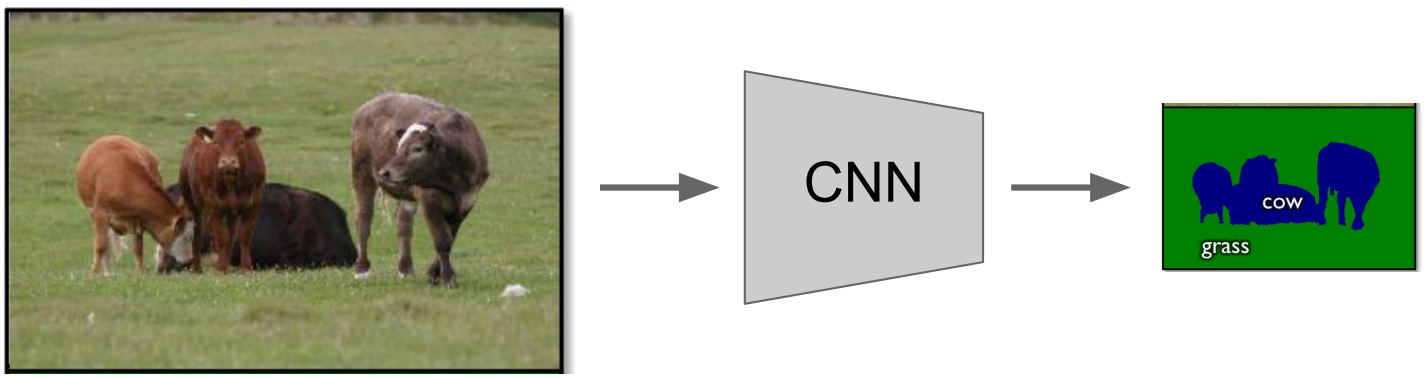


# Semantic Segmentation



# Semantic Segmentation

Run “fully convolutional” network
to get all pixels at once



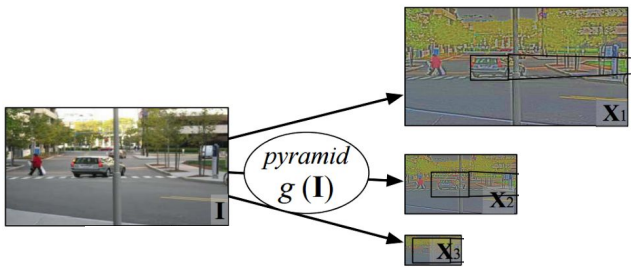

Smaller output
due to pooling

# Semantic Segmentation: Multi-Scale

Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013

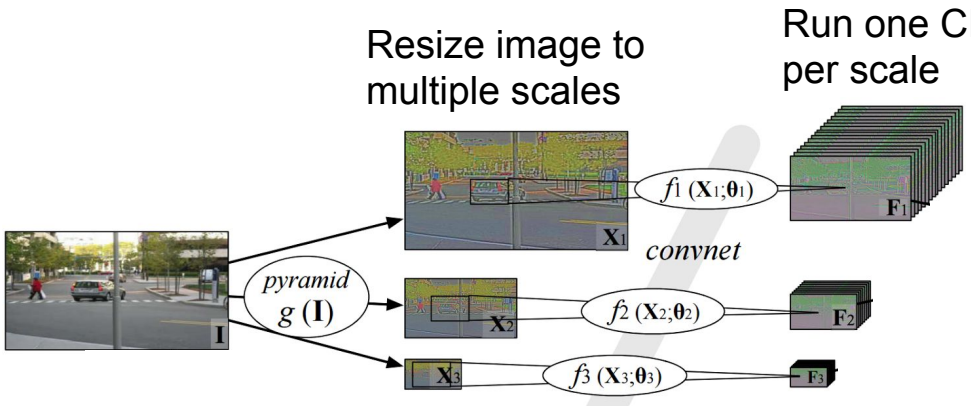
# Semantic Segmentation: Multi-Scale

Resize image to multiple scales



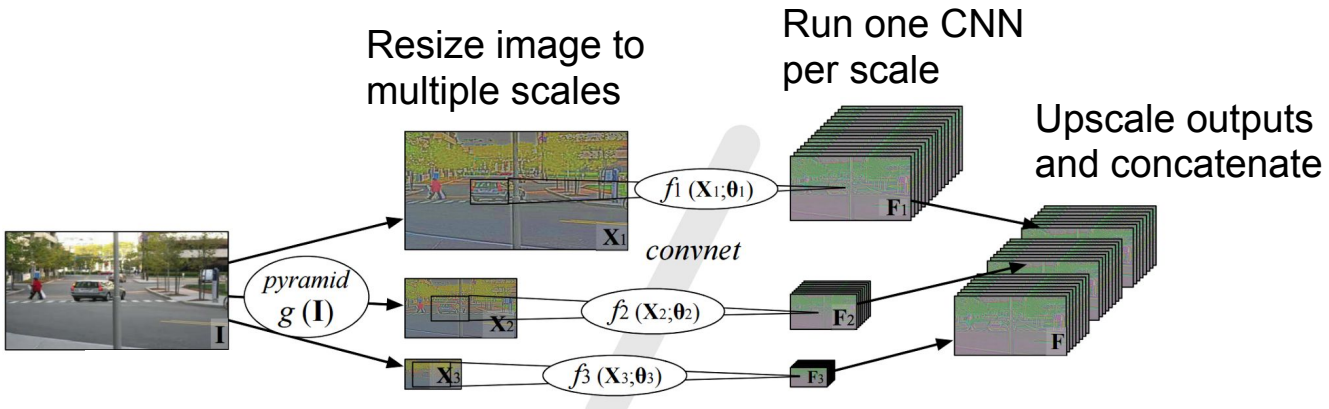

Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013

# Semantic Segmentation: Multi-Scale



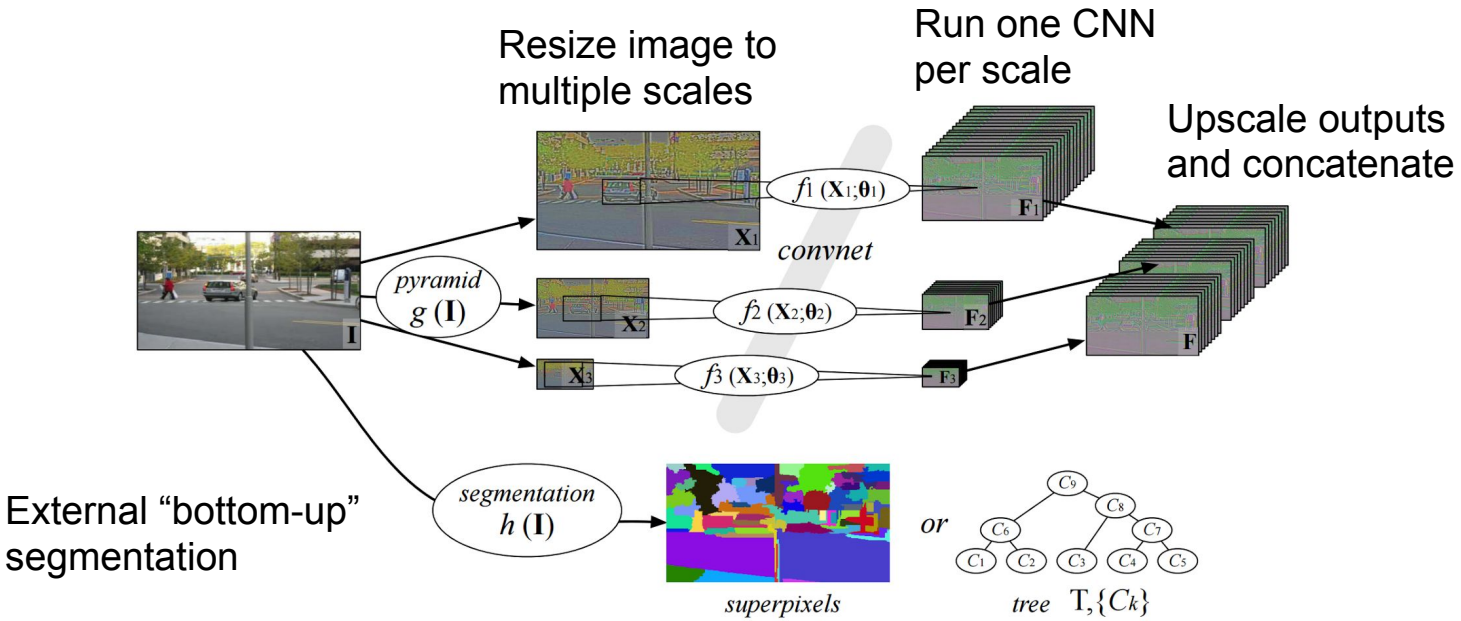

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013

# Semantic Segmentation: Multi-Scale



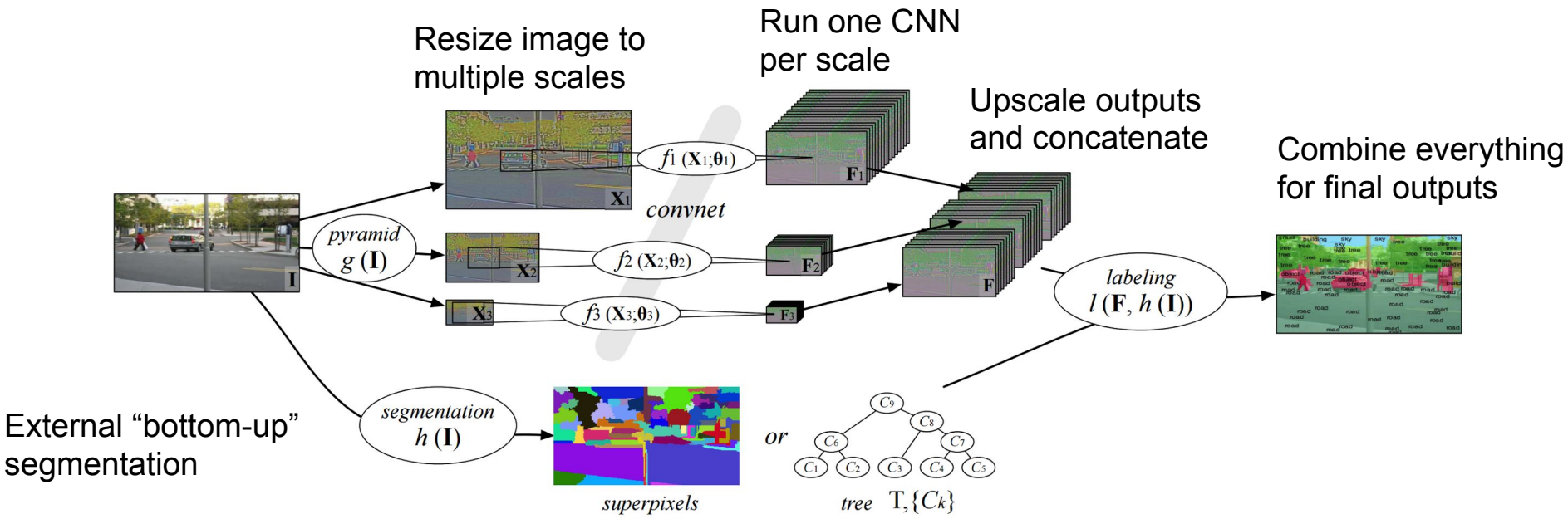

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013

# Semantic Segmentation: Multi-Scale




Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013

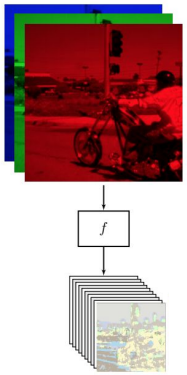
# Semantic Segmentation: Multi-Scale




Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013

# Semantic Segmentation: Refinement

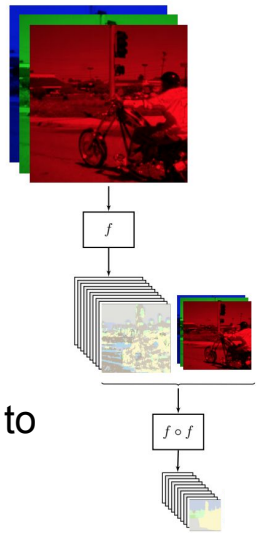
Apply CNN once
to get labels




Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

# Semantic Segmentation: Refinement

Apply CNN once to get labels




Apply AGAIN to refine labels

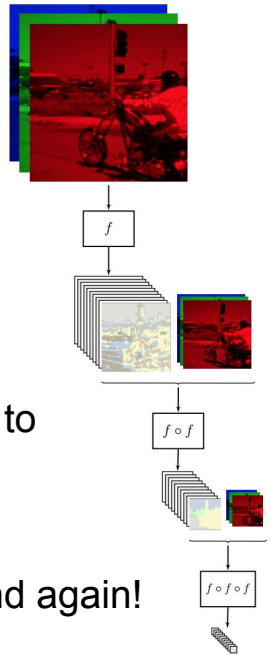
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

# Semantic Segmentation: Refinement

Apply CNN once to get labels

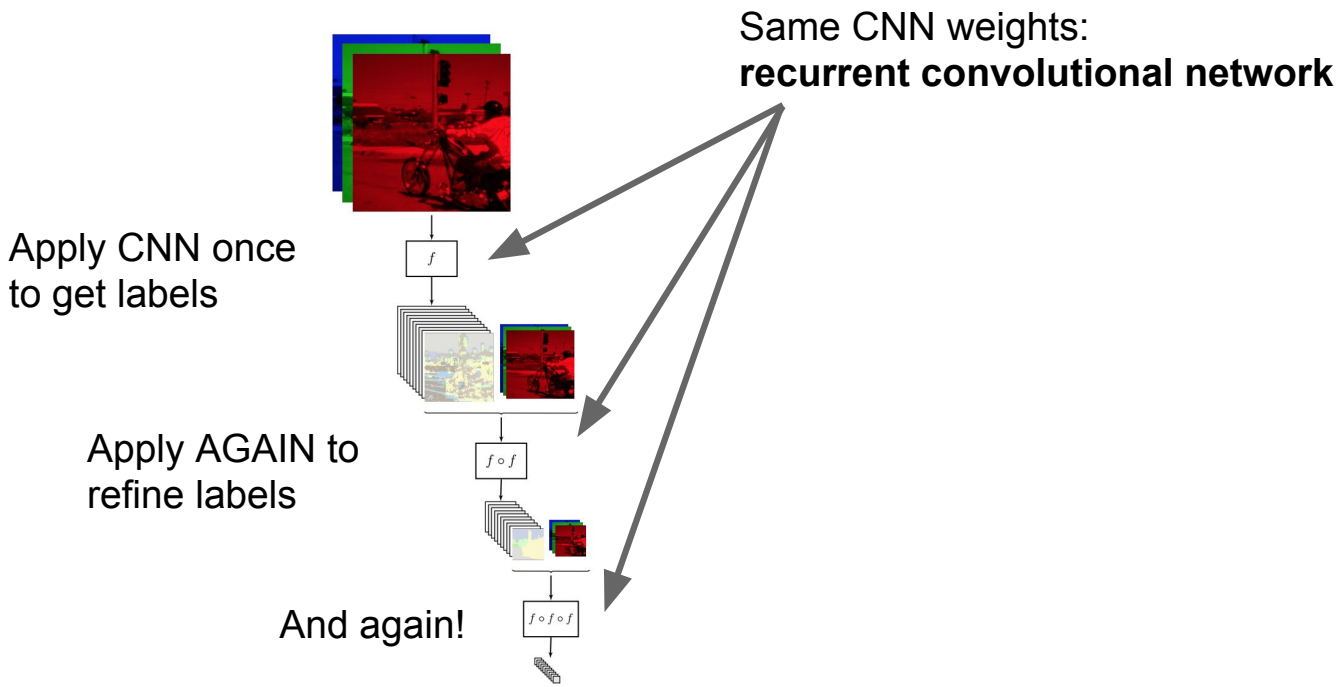
Apply AGAIN to refine labels

And again!



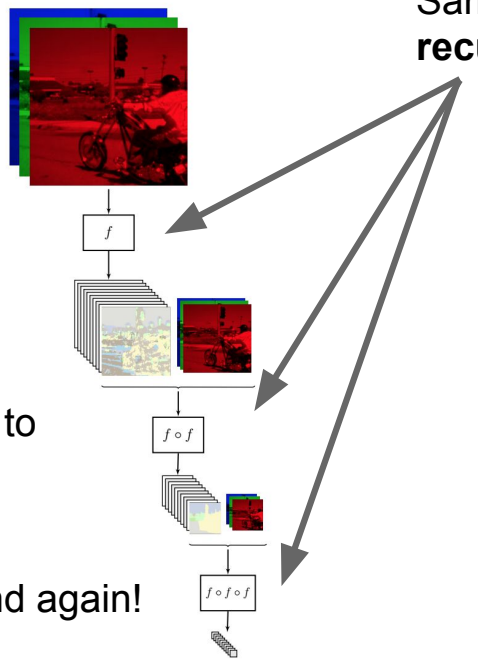

Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

# Semantic Segmentation: Refinement




Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

# Semantic Segmentation: Refinement

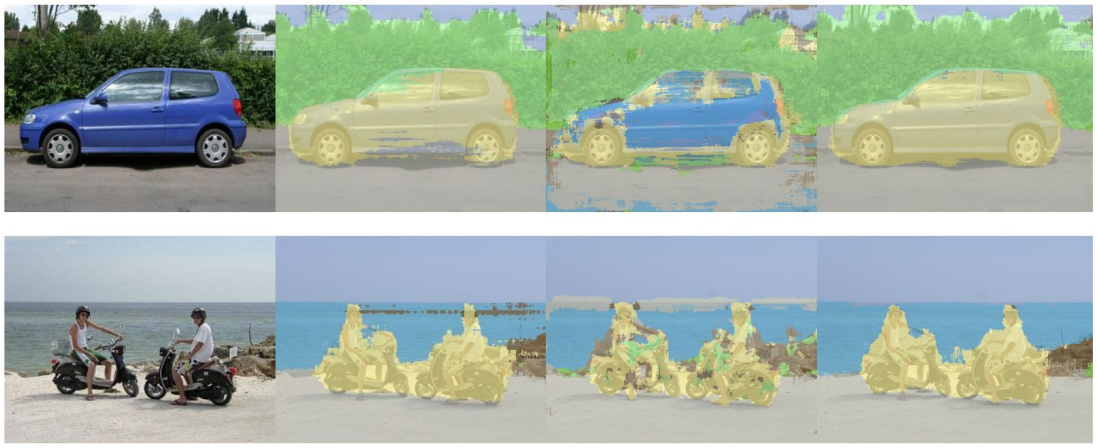



Apply CNN once to get labels

Apply AGAIN to refine labels

And again!

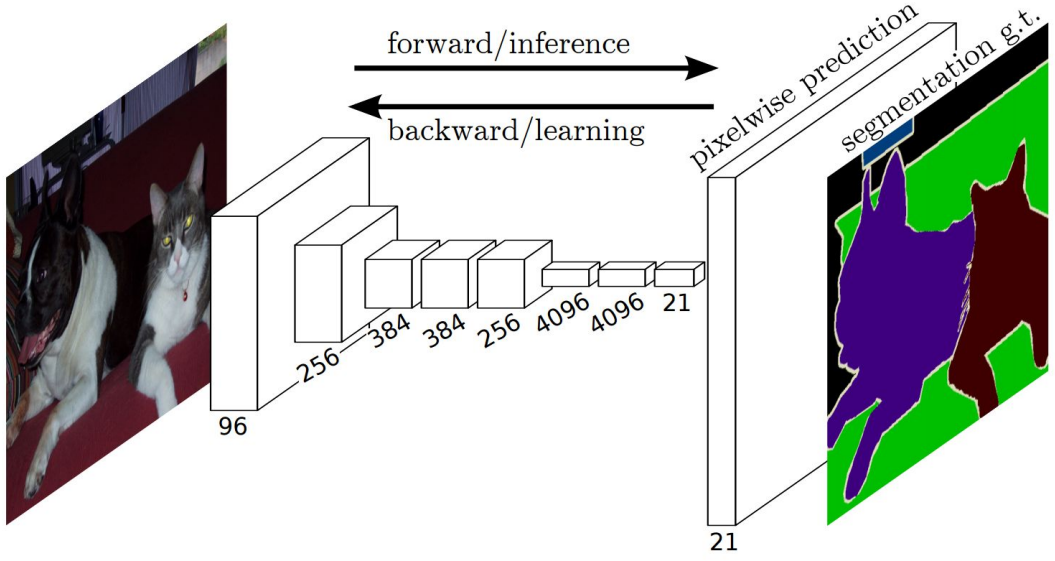
Same CNN weights:
**recurrent convolutional network**



More iterations improve results

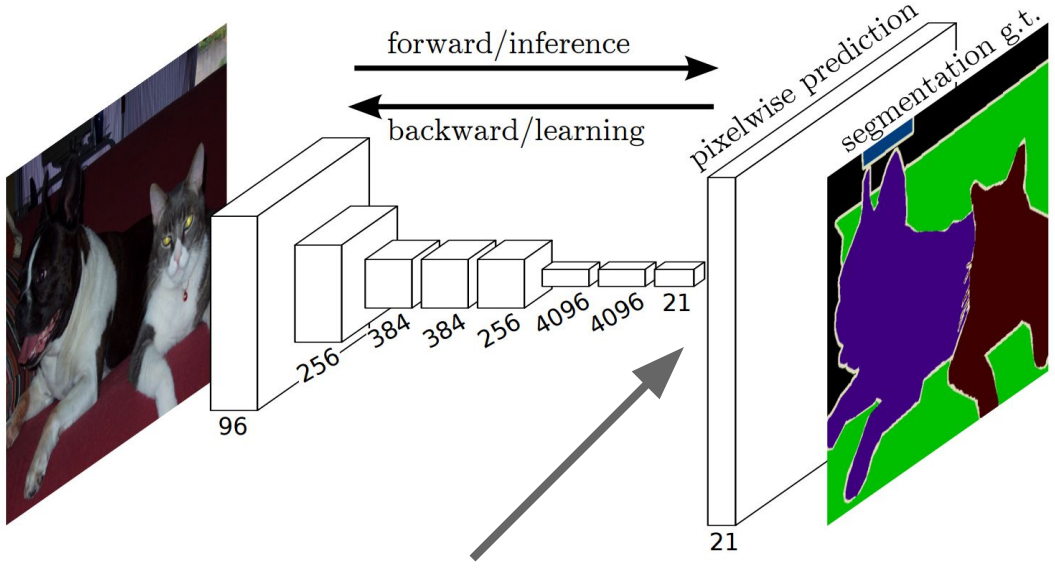
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

# Semantic Segmentation: Upsampling




Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

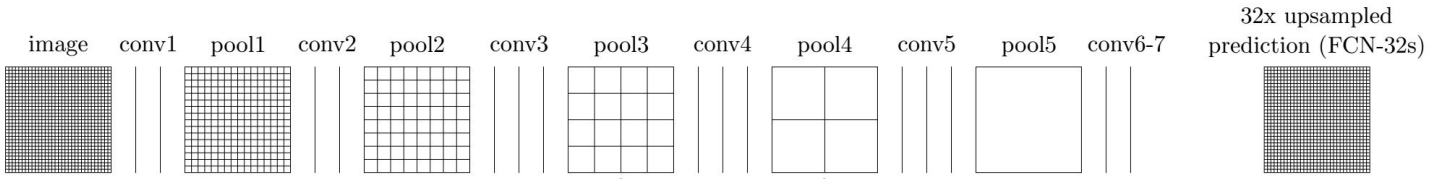
# Semantic Segmentation: Upsampling




Learnable upsampling!

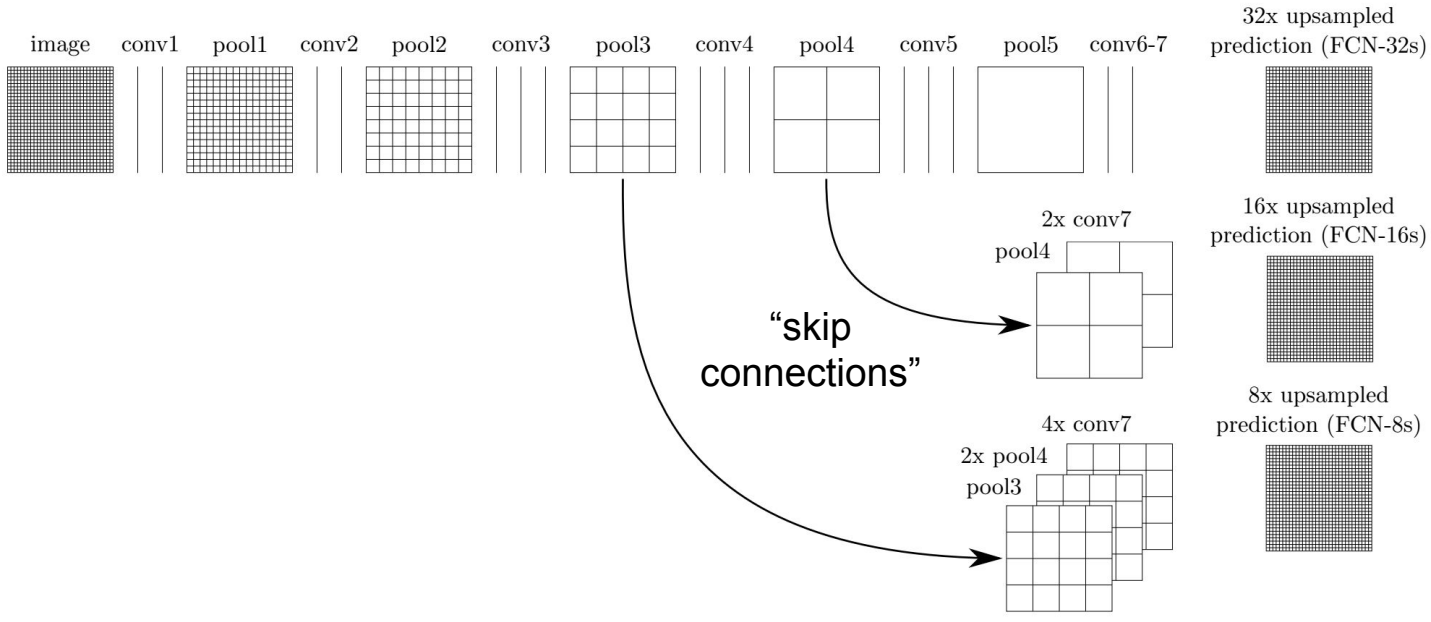
Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

# Semantic Segmentation: Upsampling



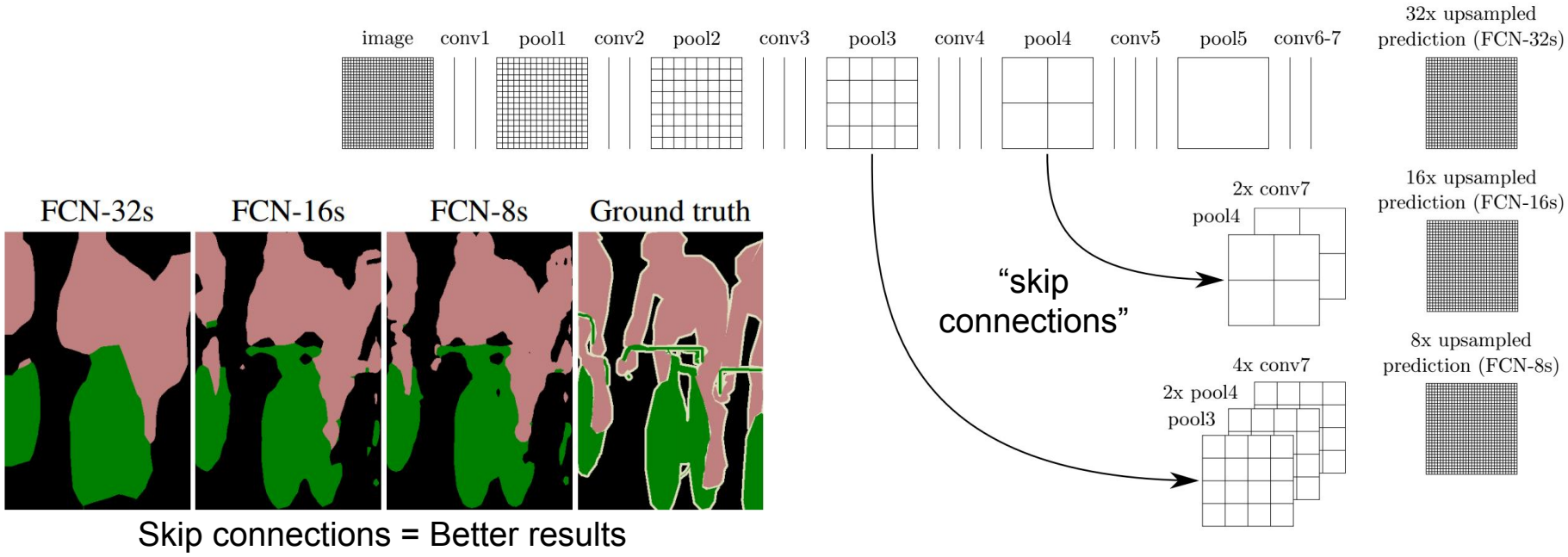

Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

# Semantic Segmentation: Upsampling




Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

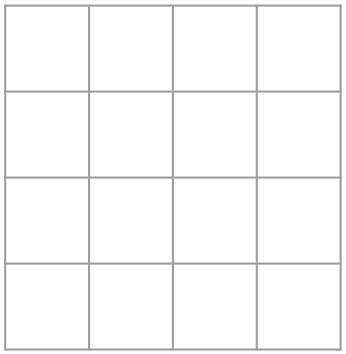
# Semantic Segmentation: Upsampling



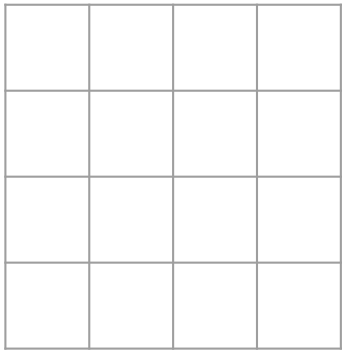

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

# Learnable Upsampling: “Deconvolution”

Typical 3 x 3 convolution, stride 1 pad 1



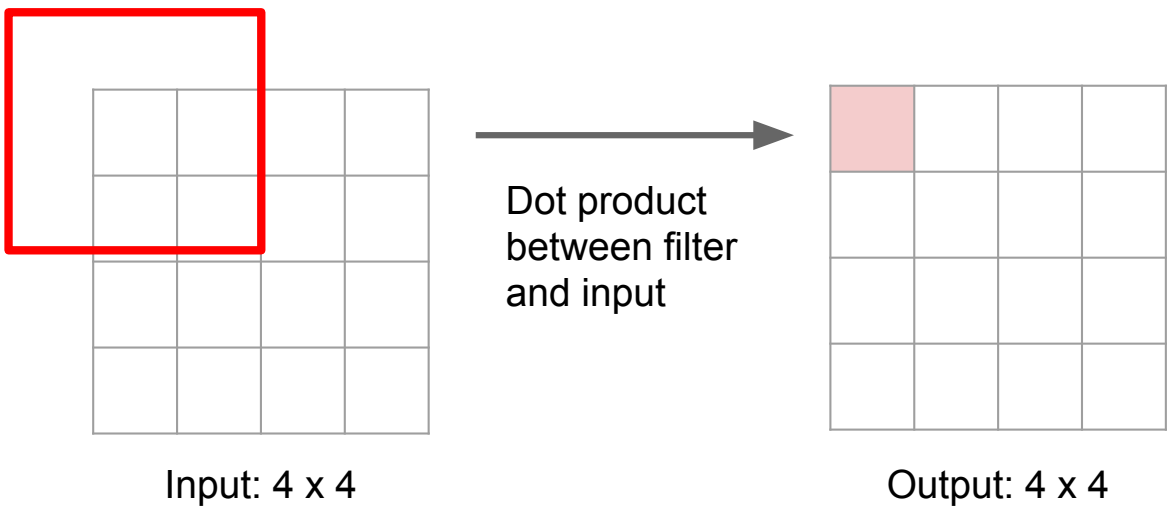
Input: 4 x 4



Output: 4 x 4

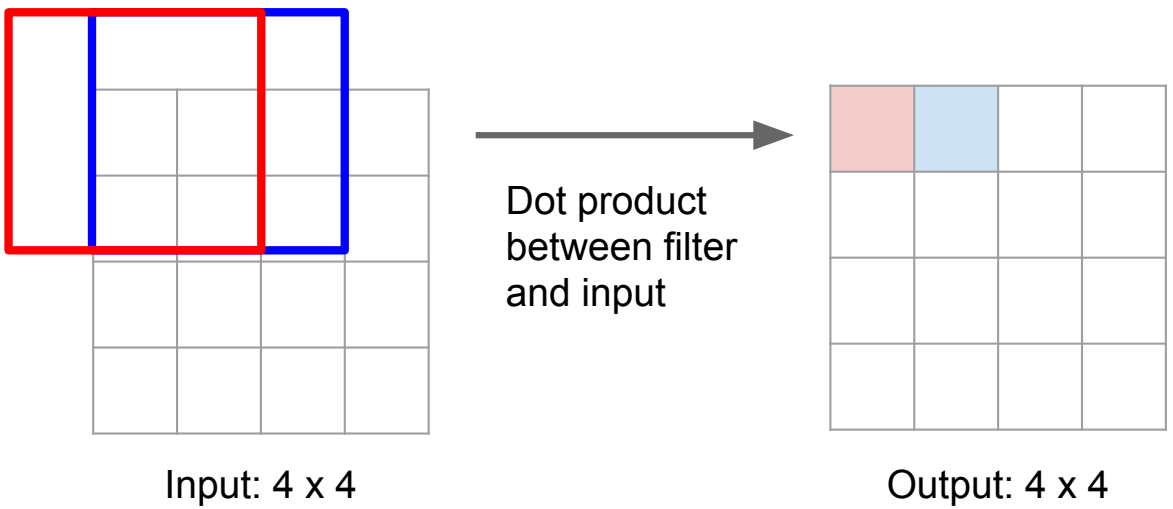
# Learnable Upsampling: “Deconvolution”

Typical 3 x 3 convolution, stride 1 pad 1



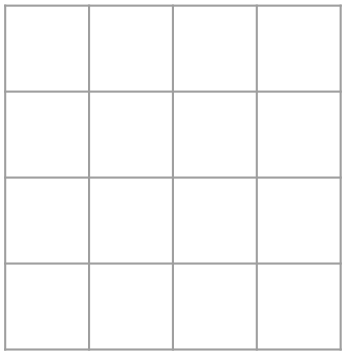
# Learnable Upsampling: “Deconvolution”

Typical 3 x 3 convolution, stride 1 pad 1

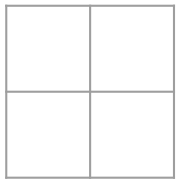


# Learnable Upsampling: “Deconvolution”

Typical 3 x 3 convolution, **stride 2** pad 1



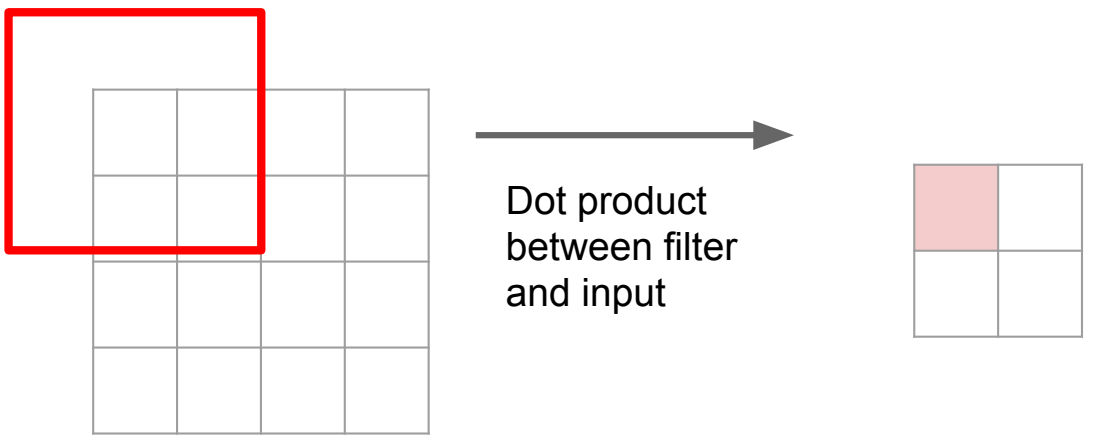
Input: 4 x 4



Output: 2 x 2

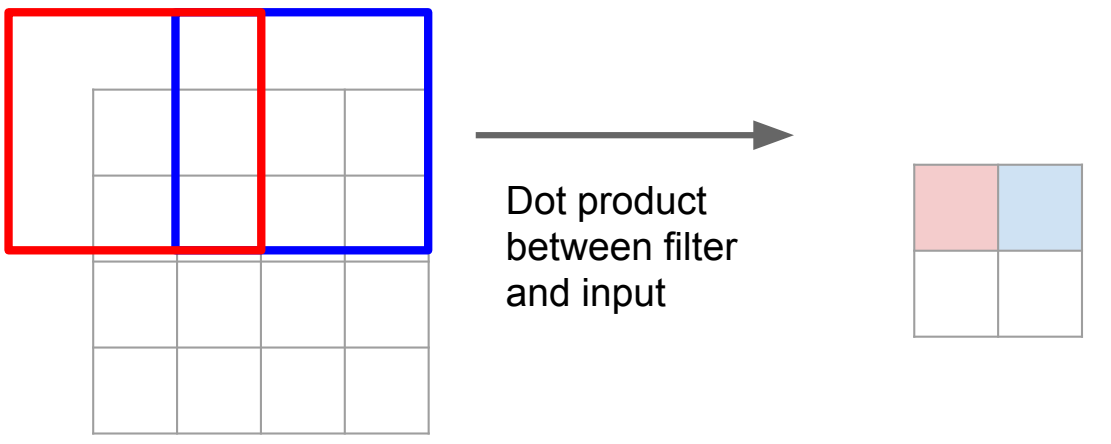
# Learnable Upsampling: “Deconvolution”

Typical 3 x 3 convolution, stride 2 pad 1



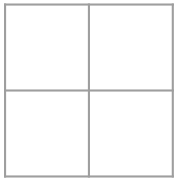
# Learnable Upsampling: “Deconvolution”

Typical 3 x 3 convolution, stride 2 pad 1

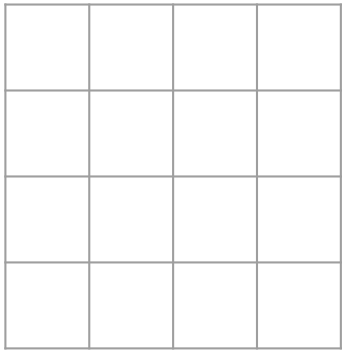


# Learnable Upsampling: “Deconvolution”

3 x 3 “deconvolution”, stride 2 pad 1



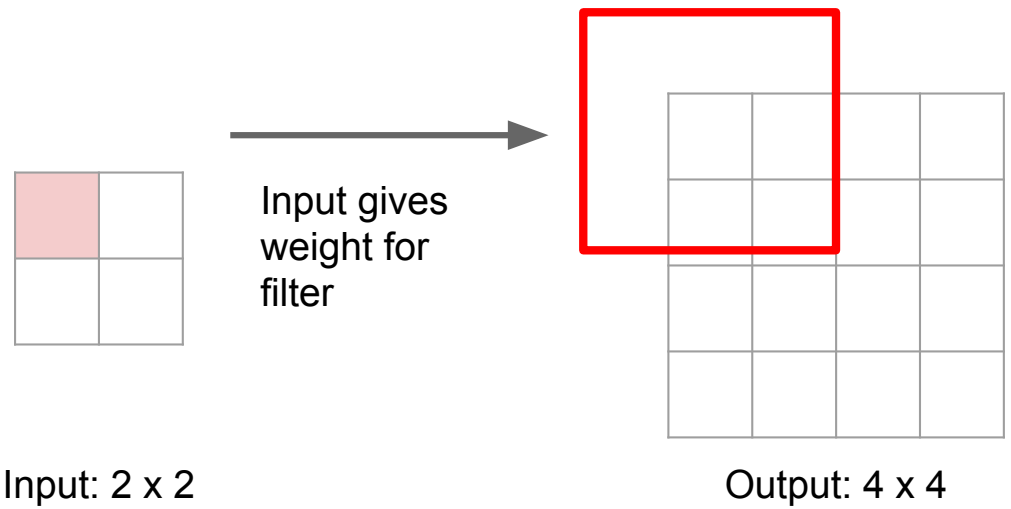
Input: 2 x 2



Output: 4 x 4

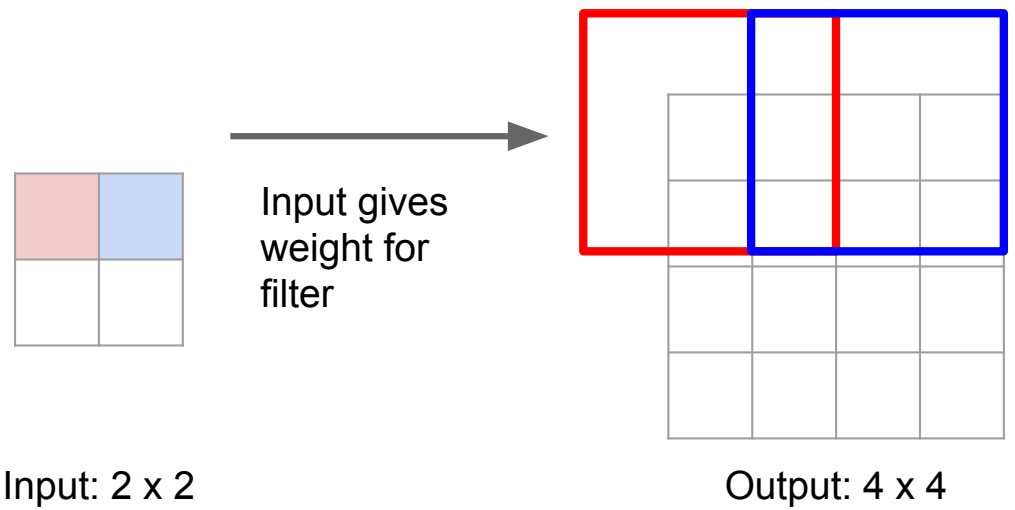
# Learnable Upsampling: “Deconvolution”

3 x 3 “deconvolution”, stride 2 pad 1

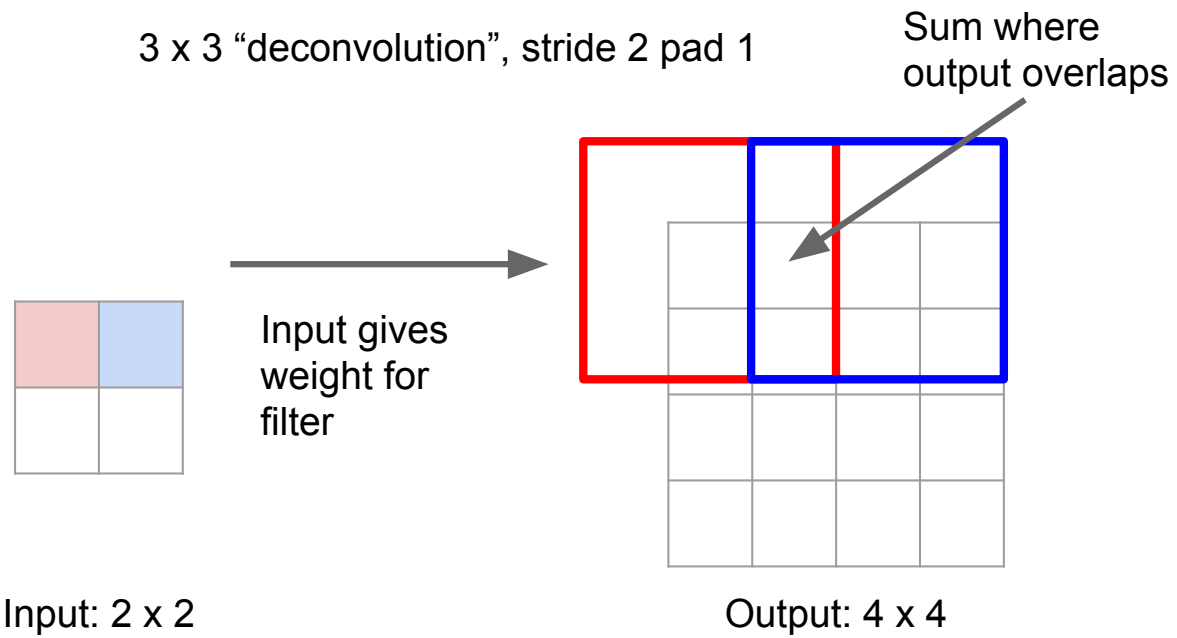


# Learnable Upsampling: “Deconvolution”

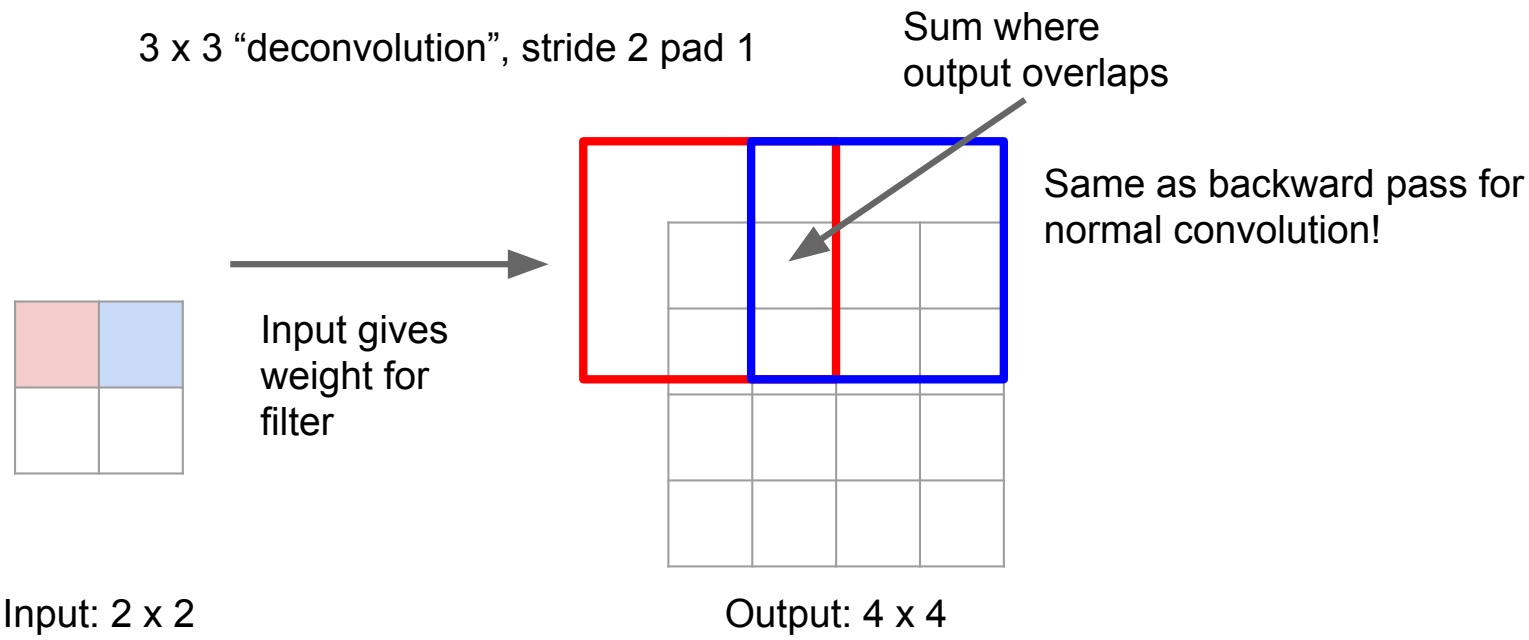
3 x 3 “deconvolution”, stride 2 pad 1



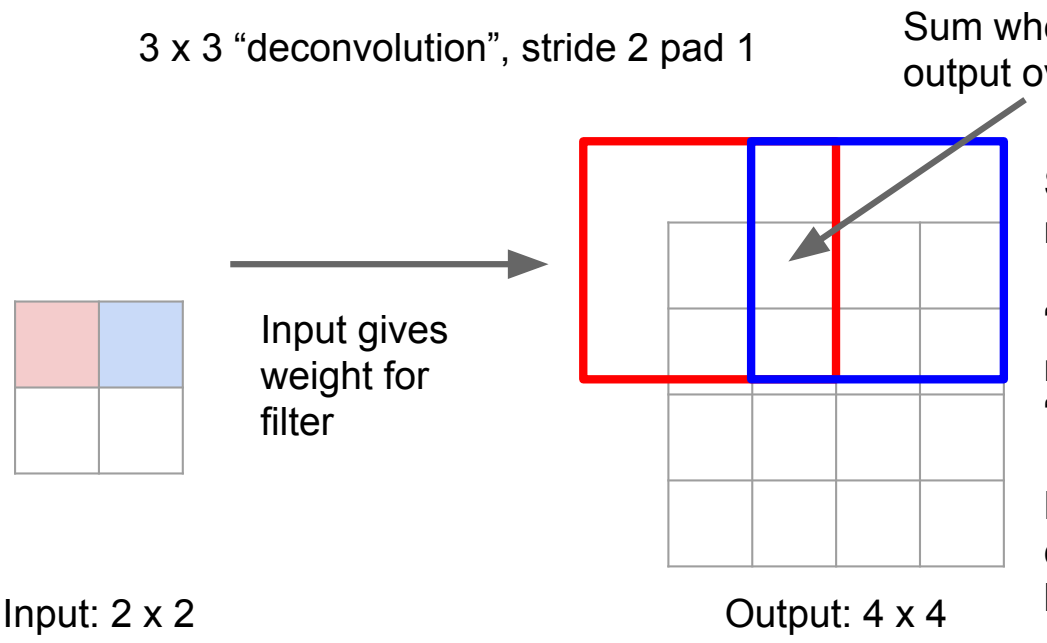
# Learnable Upsampling: “Deconvolution”



# Learnable Upsampling: “Deconvolution”



# Learnable Upsampling: “Deconvolution”




Same as backward pass for normal convolution!

“Deconvolution” is a bad name, already defined as “inverse of convolution”

**Better names:**
convolution transpose,
backward strided convolution,
1/2 strided convolution,
upconvolution

# Learnable Upsampling: “Deconvolution”

[1]It is more proper to say “convolutional transpose operation” rather than “deconvolutional” operation. Hence, we will be using the term “convolutional transpose” from now.

Im et al, “Generating images with recurrent adversarial networks”, arXiv 2016

A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions)

Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

“Deconvolution” is a bad name, already defined as “inverse of convolution”

**Better names:**
convolution transpose,
backward strided convolution,
1/2 strided convolution,
upconvolution

# Learnable Upsampling: “Deconvolution”

Great explanation
in appendix

[1]It is more proper to say “convolutional transpose operation” rather than “deconvolutional” operation. Hence, we will be using the term “convolutional transpose” from now.

Im et al, “Generating images with recurrent adversarial networks”, arXiv 2016

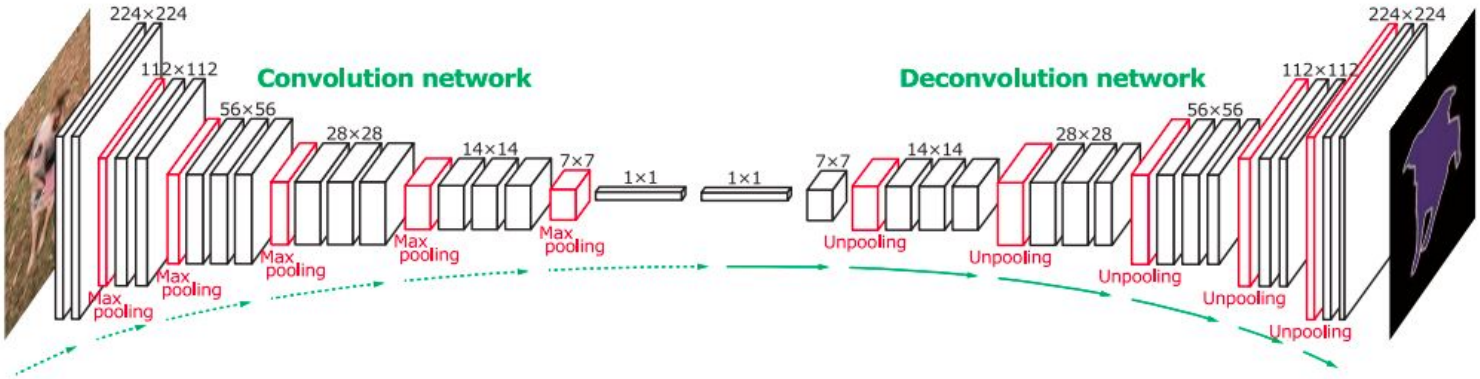
A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions)

Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

“Deconvolution” is a bad name, already defined as “inverse of convolution”

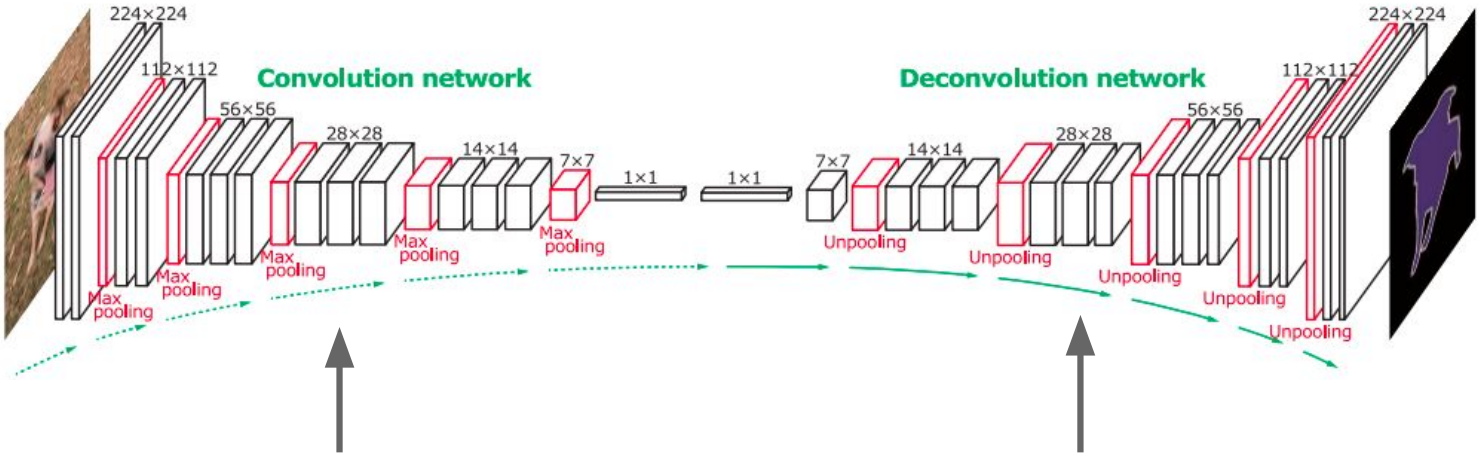
**Better names:**
convolution transpose,
backward strided convolution,
1/2 strided convolution,
upconvolution

# Semantic Segmentation: Upsampling




Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

# Semantic Segmentation: Upsampling




Normal VGG

"Upside down" VGG

Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

6 days of training on Titan X…

# Attention Models

# RNNs for text translation

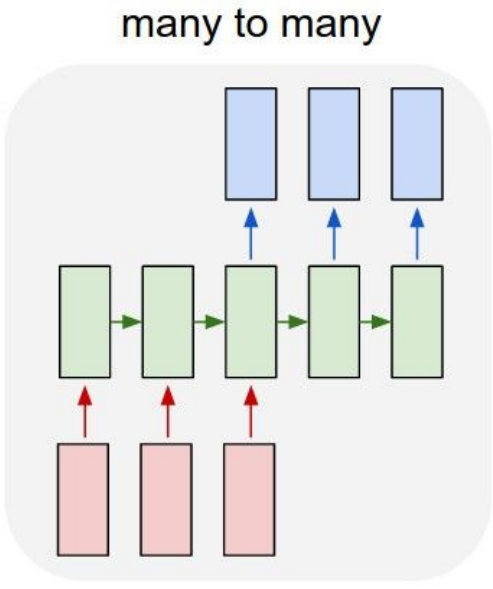
- **Encode** sentence in with one RNN.
- Then **Decode** the sentence with another RNN...

Focus on:

**Neural Machine Translation by Jointly Learning to Align and Translate** by Bahdanau et al, ICLR 2015.

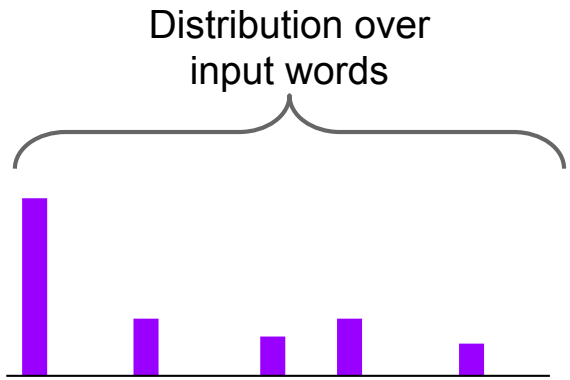
# Soft Attention for Translation

“Mi gato es el mejor” -> “My cat is the best”

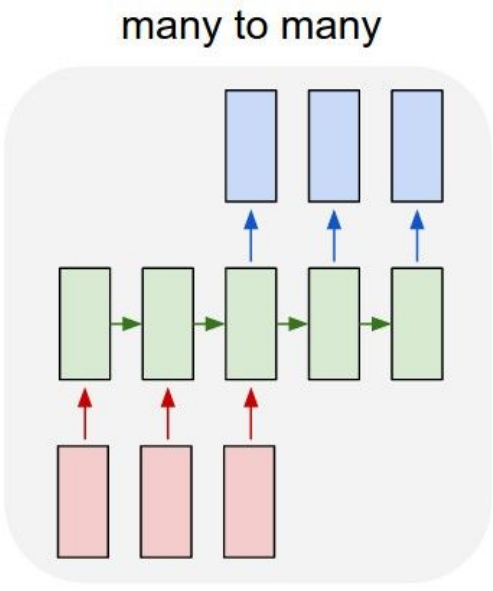



Bahdanau et al, “Neural Machine Translation by Jointly Learning to Align and Translate”, ICLR 2015

# Soft Attention for Translation



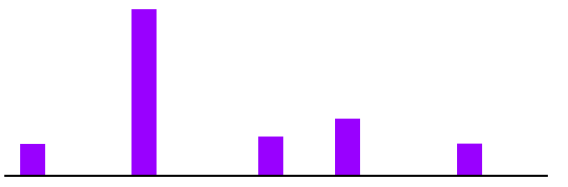

“Mi gato es el mejor” -> “My cat is the best”



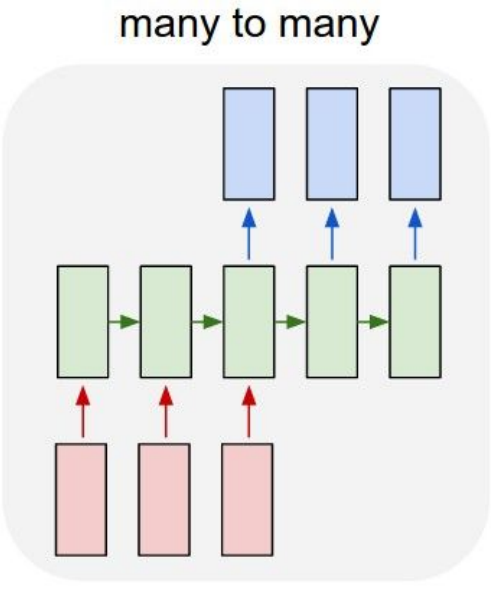

Bahdanau et al, “Neural Machine Translation by Jointly Learning to Align and Translate”, ICLR 2015

# Soft Attention for Translation

Distribution over input words

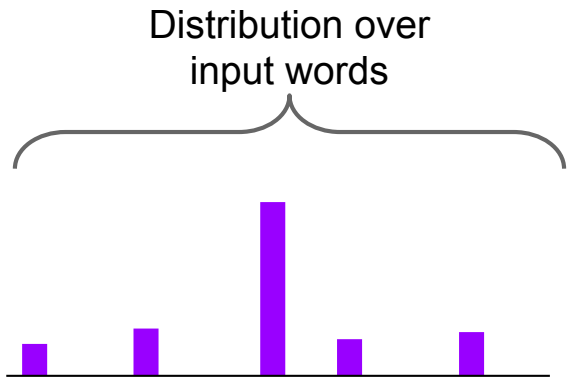


“Mi gato es el mejor” -> “My cat is the best”

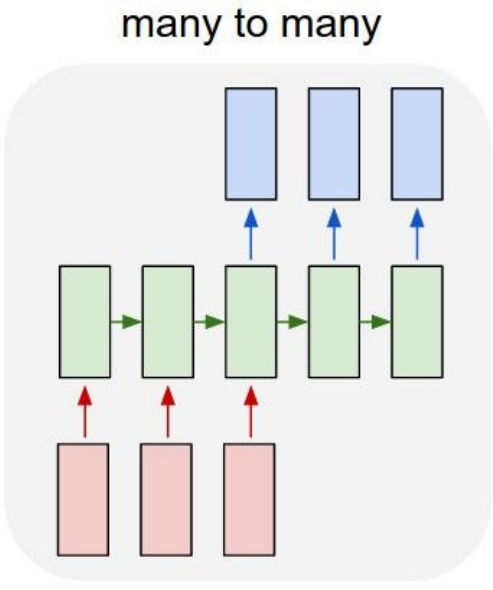



Bahdanau et al, “Neural Machine Translation by Jointly Learning to Align and Translate”, ICLR 2015

# Soft Attention for Translation




“Mi gato es el mejor” -> “My cat is the best”




Bahdanau et al, “Neural Machine Translation by Jointly Learning to Align and Translate”, ICLR 2015

# Example effect of search

Sample result from:

**Neural Machine Translation by Jointly Learning to Align and Translate** by Bahdanau et al, ICLR 2015.

- Test sentence in English:

  *An admitting privilege is the right of a doctor to admit a patient to a hospital or a medical centre to carry out a diagnosis or a procedure, based on his status as a health care worker at a hospital.*

- Translation with **no** attention mechanism:

  *Un privilège d'admission est le droit d'un médecin de reconnaître un patient à l'hôpital ou un centre médical d'un diagnostic ou de prendre un diagnostic en fonction de son état de santé.*

- Translation with attention mechanism:

  *Un privilège d'admission est le droit d'un médecin d'admettre un patient à un hôpital ou un centre médical pour effectuer un diagnostic ou une procédure, selon son statut de travailleur des soins de santé à l'hôpital.*

# Example effect of search

Sample result from:

**Neural Machine Translation by Jointly Learning to Align and Translate** by Bahdanau et al, ICLR 2015.

- Test sentence in English:

  *An admitting privilege is the right of a doctor to admit a patient to a hospital or a medical centre to carry out a diagnosis or a procedure, based on his status as a health care worker at a hospital.*

- Translation with **no** attention mechanism:

  *Un privilège d'admission est le droit d'un médecin de reconnaître un patient à l'hôpital ou un centre médical d'un diagnostic ou de prendre un diagnostic en fonction de son état de santé.*

- Translation with attention mechanism:

  *Un privilège d'admission est le droit d'un médecin d'admettre un patient à un hôpital ou un centre médical pour effectuer un diagnostic ou une procédure, selon son statut de travailleur des soins de santé à l'hôpital.*

Can the French speakers confirm this is a better translation?

# ConvNets & RNNs for image captioning

- **Encode** image in with a ConvNet.
- Then **Decode** the image with a RNN.

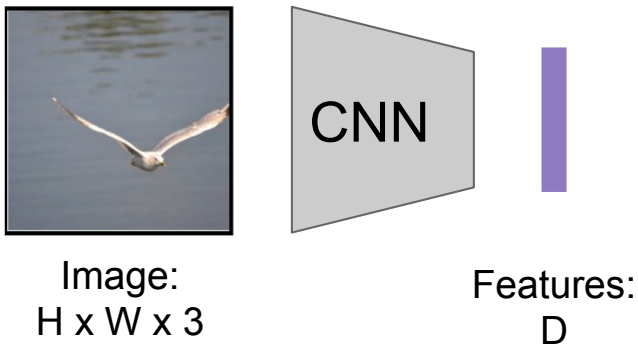
Focus on:

**Show, Attend and Tell: Neural Image Caption Generation with Visual Attention** by Xu et al, ICML 2015

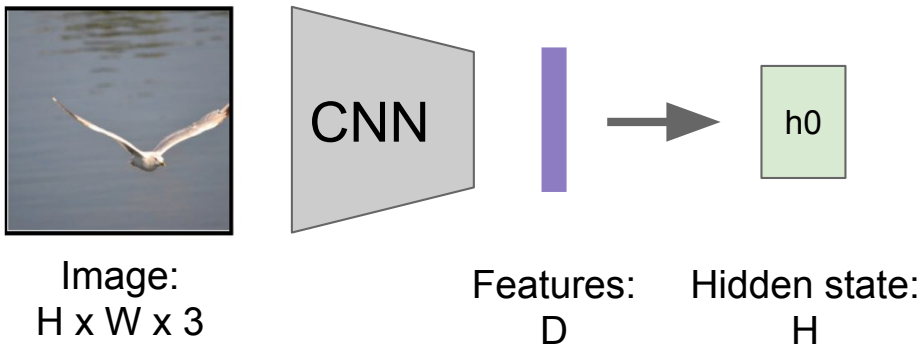
# Recall: RNN for Captioning

Image:
H x W x 3

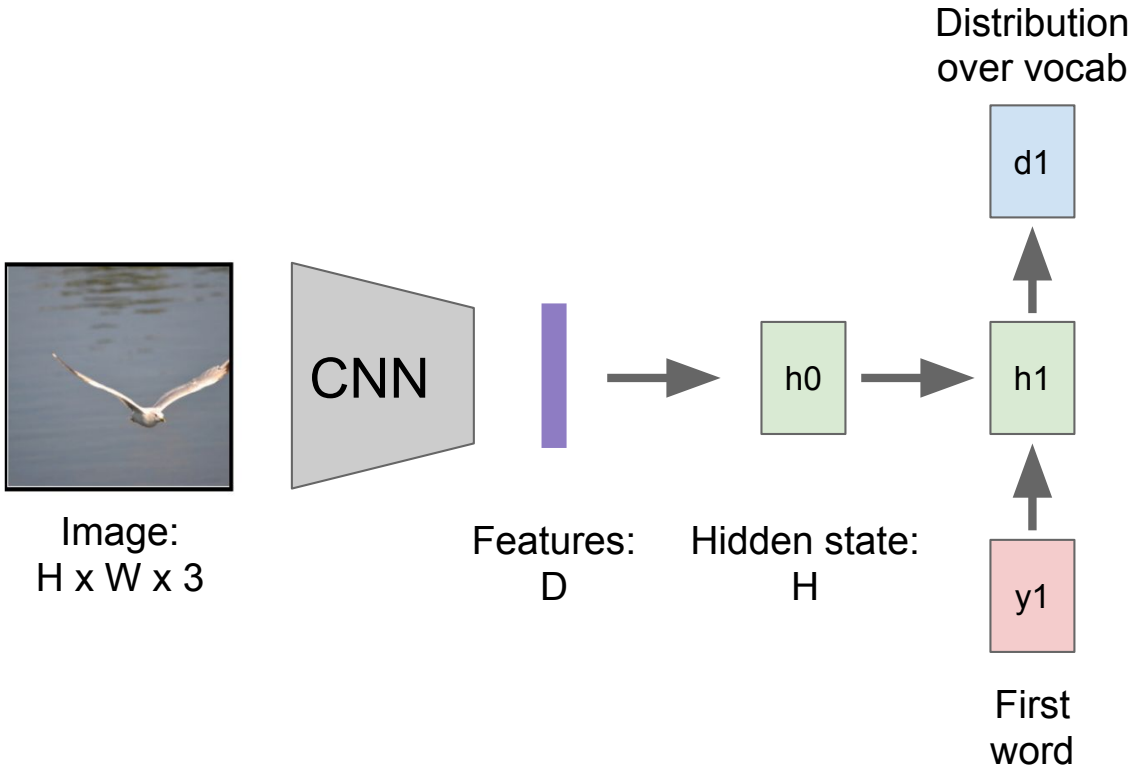
# Recall: RNN for Captioning



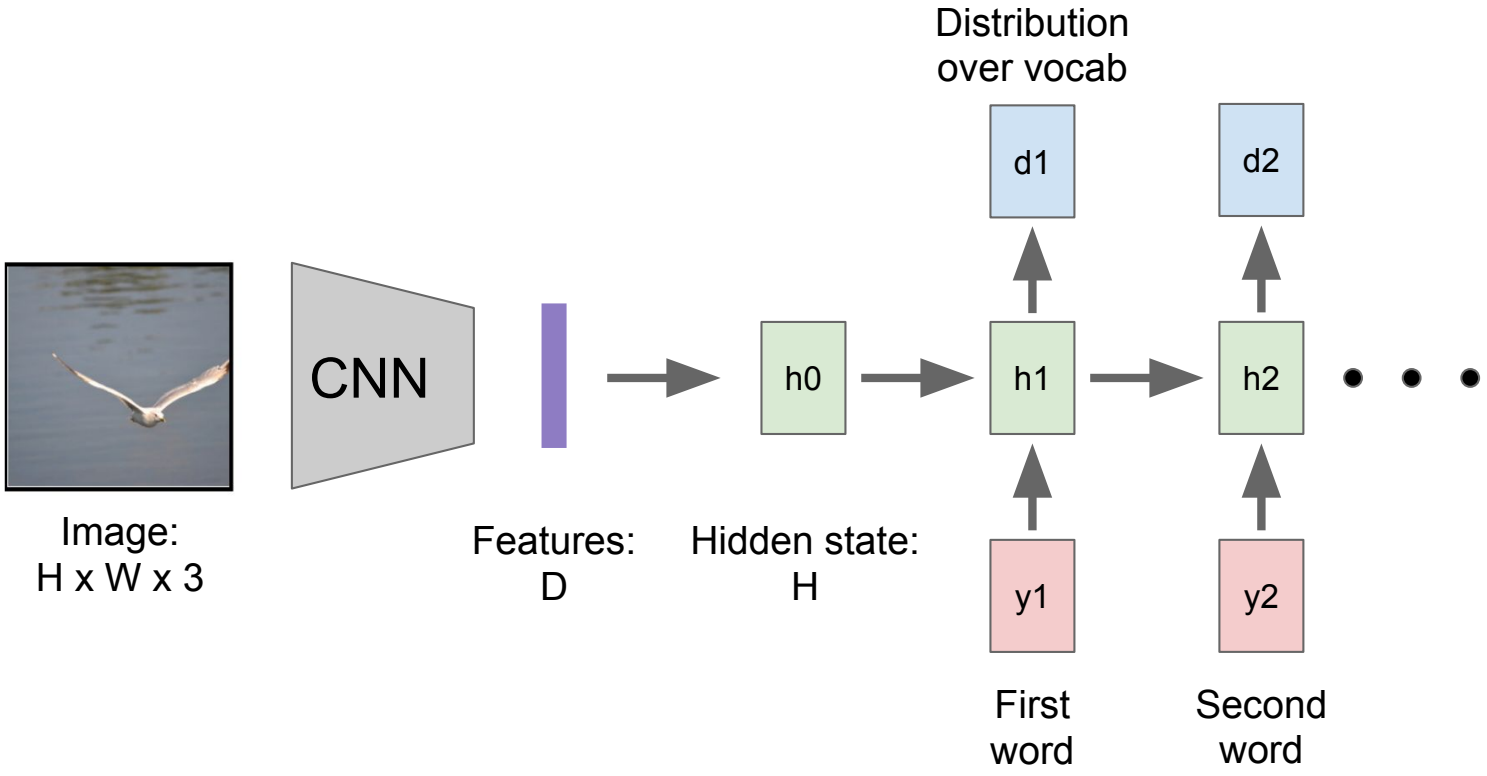
# Recall: RNN for Captioning



# Recall: RNN for Captioning

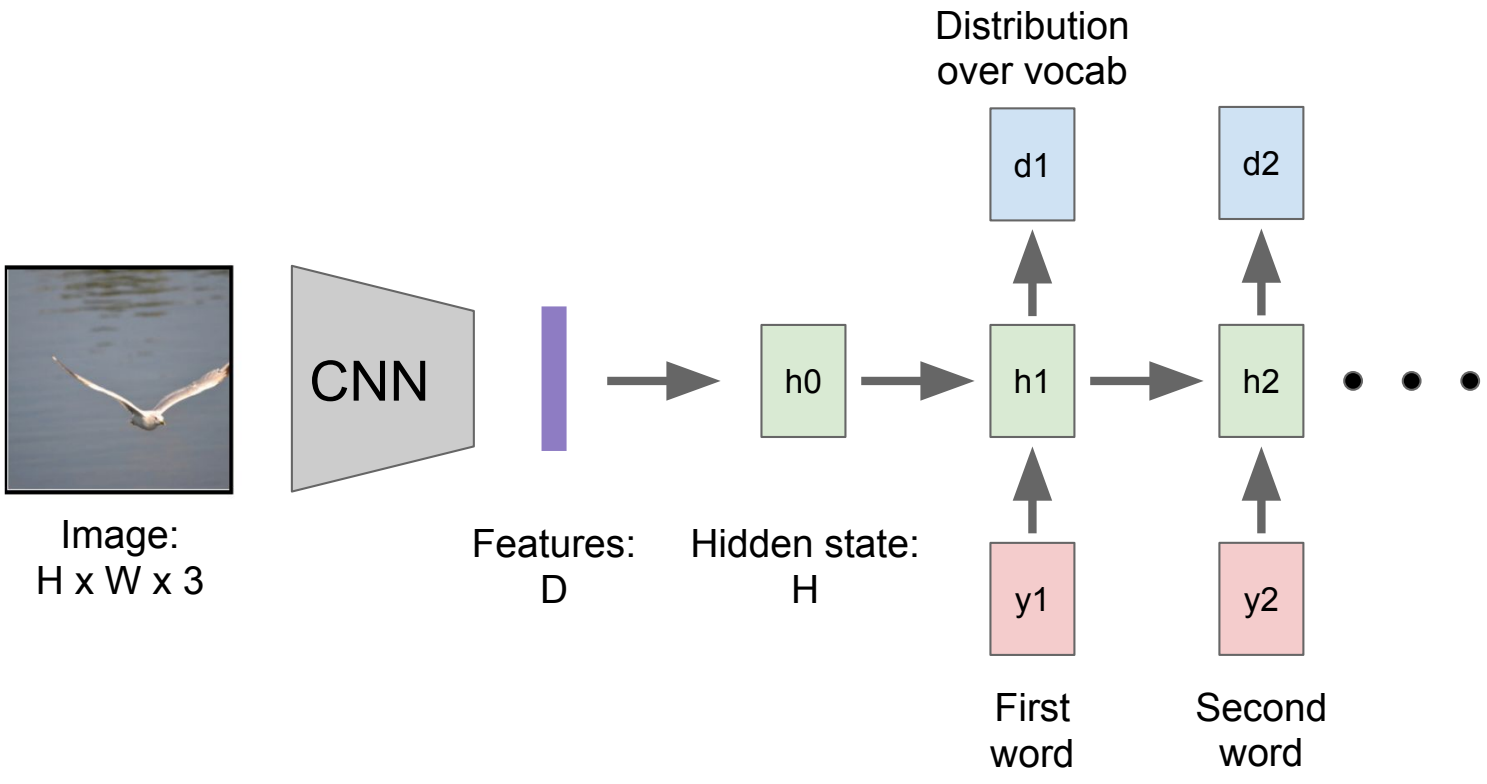


# Recall: RNN for Captioning

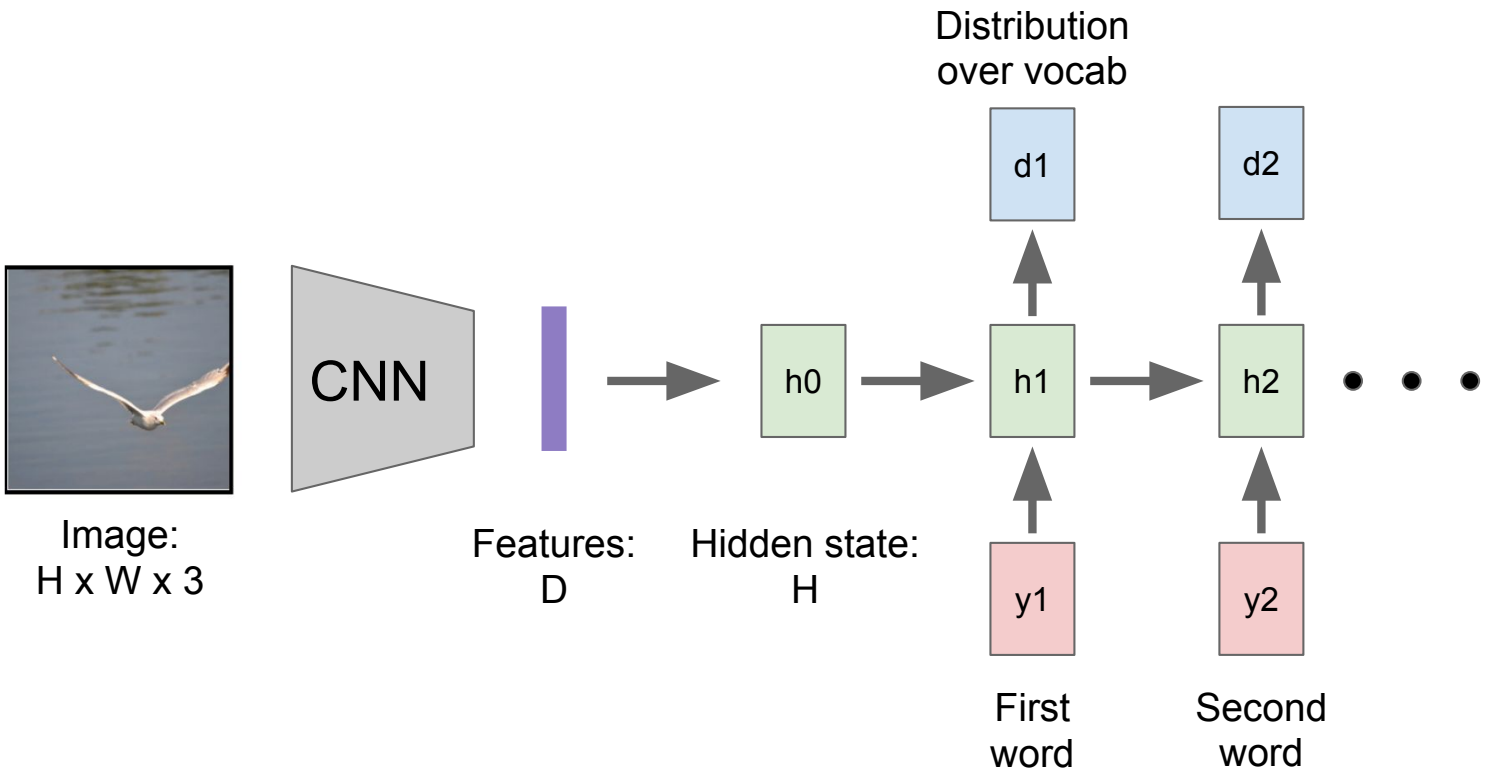


# Recall: RNN for Captioning

RNN only looks at whole image, once

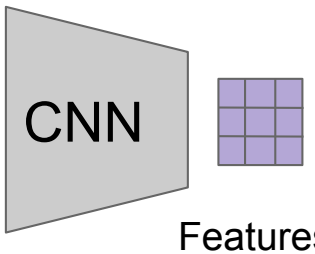


# Recall: RNN for Captioning



# Soft Attention for Captioning

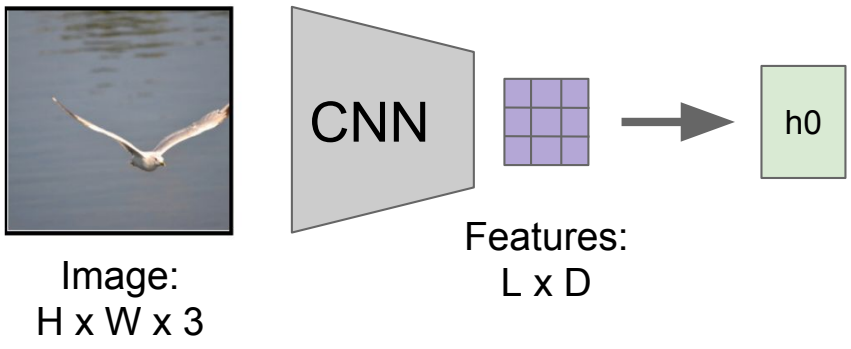
Image:
H x W x 3




Features:
L x D

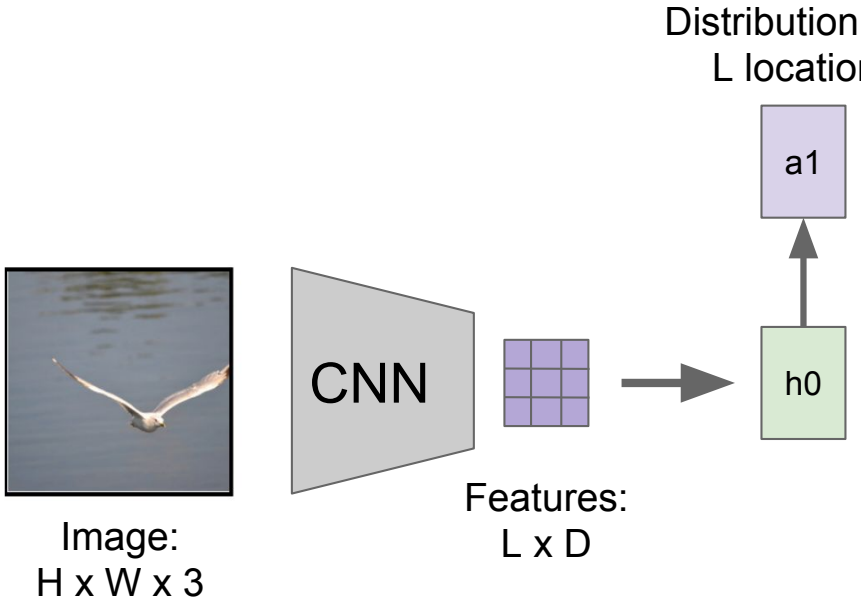
Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

# Soft Attention for Captioning



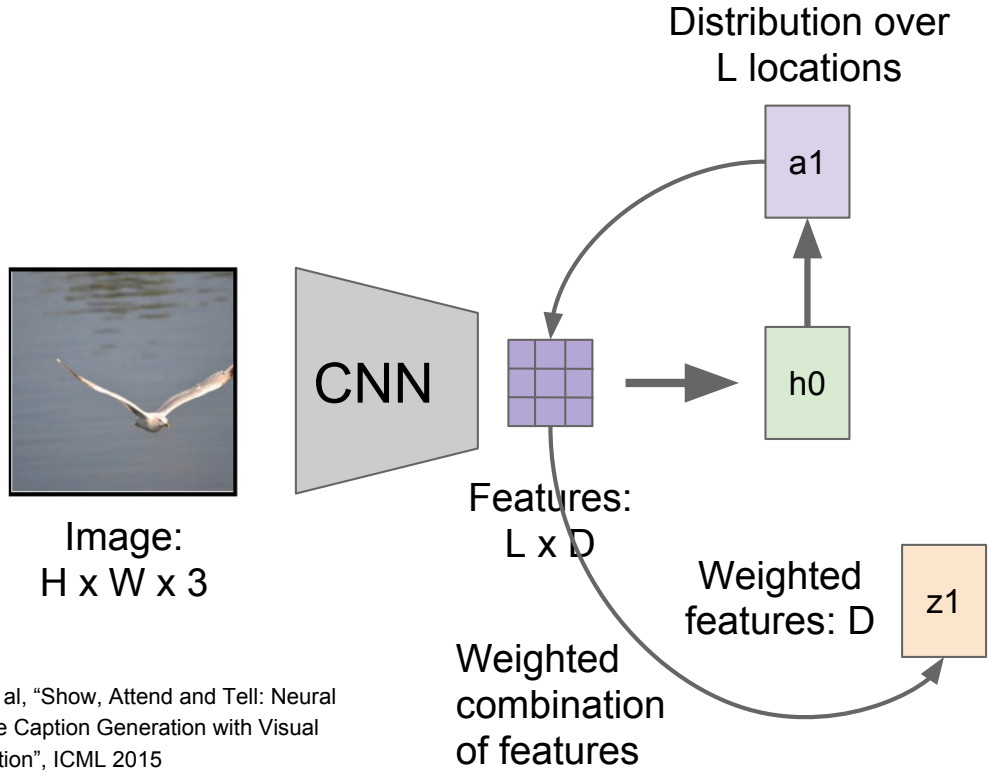

Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

# Soft Attention for Captioning



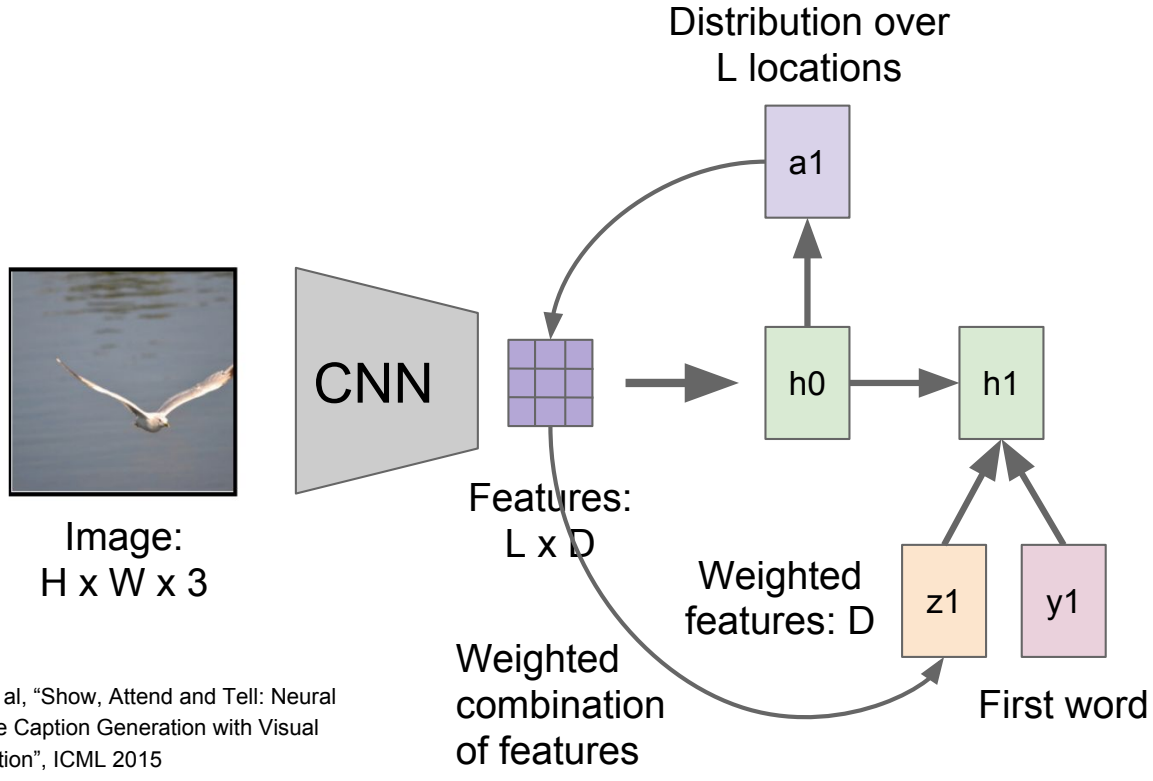

Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

# Soft Attention for Captioning



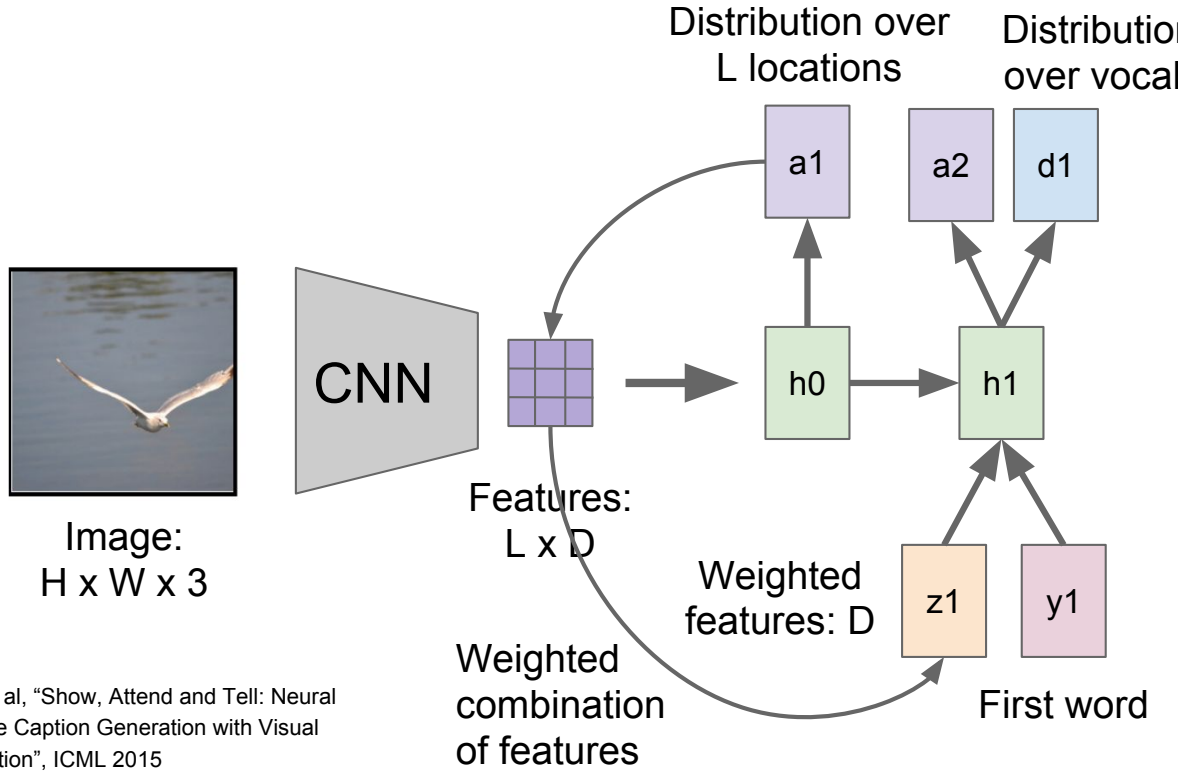

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Soft Attention for Captioning



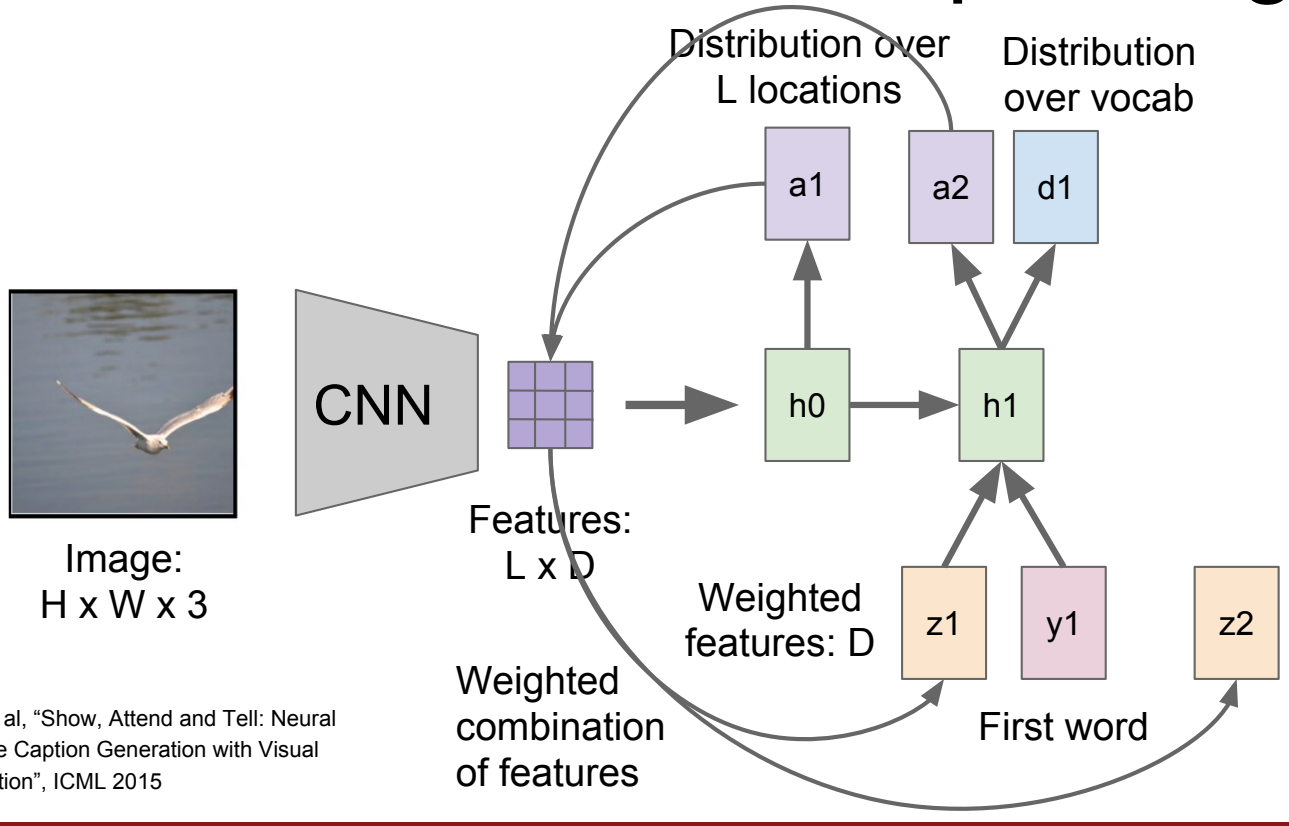

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Soft Attention for Captioning



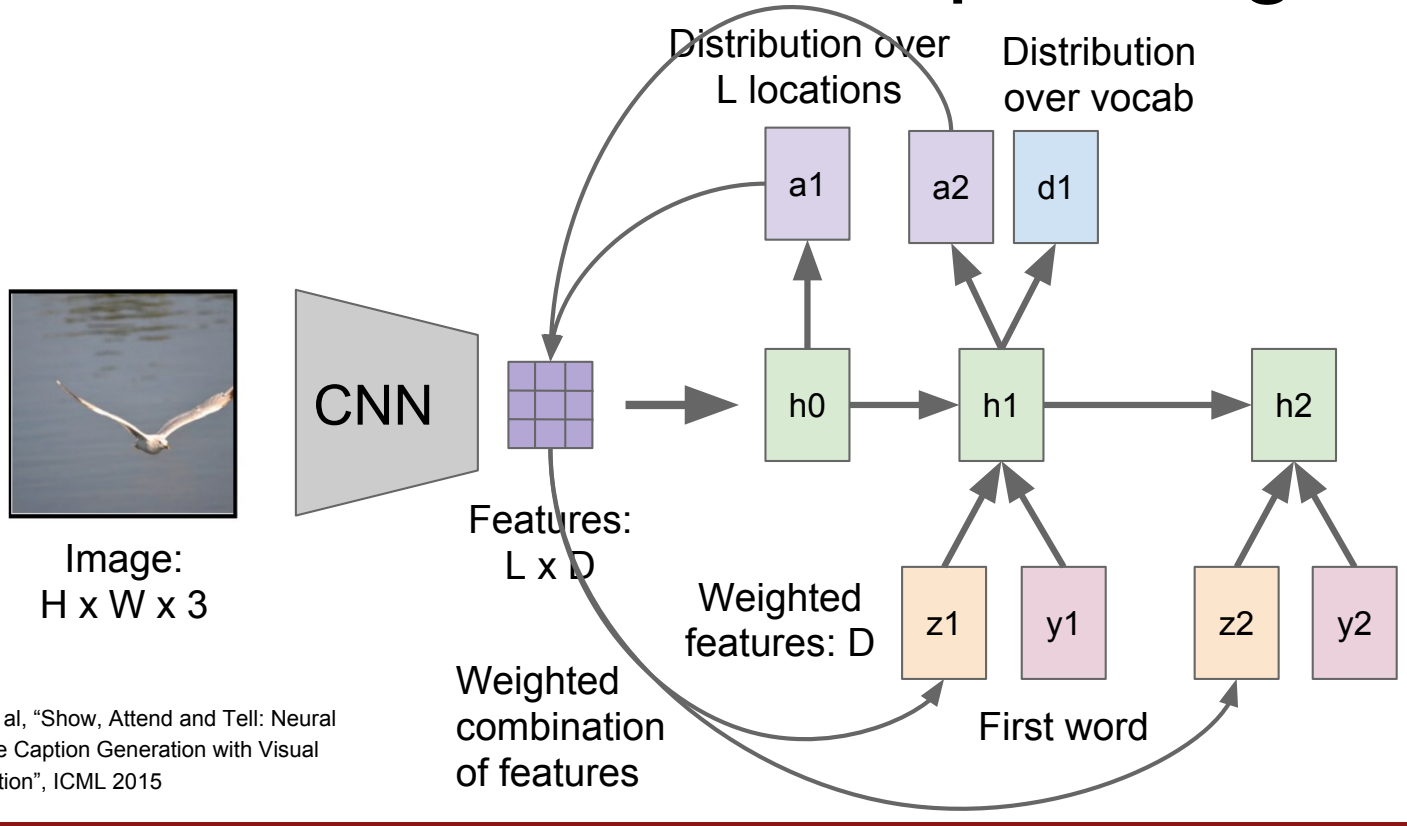

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Soft Attention for Captioning



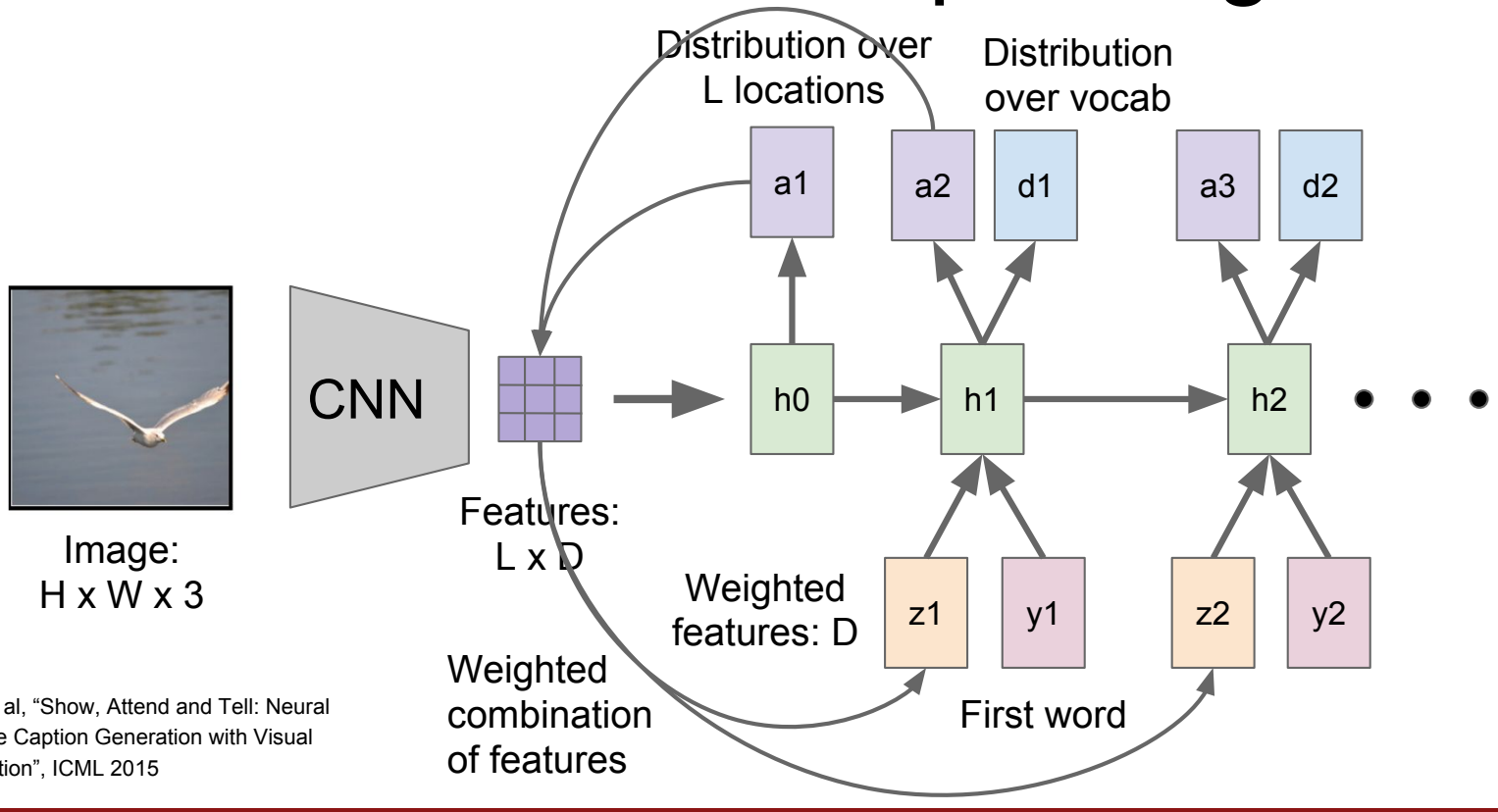

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Soft Attention for Captioning




Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Soft Attention for Captioning




Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Soft vs Hard Attention

Image:
H x W x 3

CNN

| | |
|---|---|
| a | b |
| c | d |

Grid of features
(Each D-dimensional)

From RNN:

| | |
|---|---|
| $p_a$ | $p_b$ |
| $p_c$ | $p_d$ |

Distribution over grid locations

$p_a + p_b + p_c + p_c = 1$

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Soft vs Hard Attention

Image:
H x W x 3

CNN

| | |
|---|---|
| a | b |
| c | d |

Grid of features
(Each D-dimensional)

Context vector z
(D-dimensional)

From RNN:

| | |
|---|---|
| $p_a$ | $p_b$ |
| $p_c$ | $p_d$ |

Distribution over grid locations

$p_a + p_b + p_c + p_c = 1$

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Soft vs Hard Attention

Image:
H x W x 3

CNN

| | |
|---|---|
| a | b |
| c | d |

Grid of features
(Each D-dimensional)

Context vector z
(D-dimensional)

From RNN:

| | |
|---|---|
| $p_a$ | $p_b$ |
| $p_c$ | $p_d$ |

Distribution over grid locations
$p_a + p_b + p_c + p_c = 1$

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

**Soft attention:**
Summarize ALL locations
$z = p_a a + p_b b + p_c c + p_d d$

Derivative dz/dp is nice!
Train with gradient descent

# Soft vs Hard Attention

Image:
H x W x 3

CNN

| | |
|---|---|
| a | b |
| c | d |

Grid of features
(Each D-dimensional)

Context vector z
(D-dimensional)

From RNN:

| | |
|---|---|
| $p_a$ | $p_b$ |
| $p_c$ | $p_d$ |

Distribution over grid locations
$p_a + p_b + p_c + p_c = 1$

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

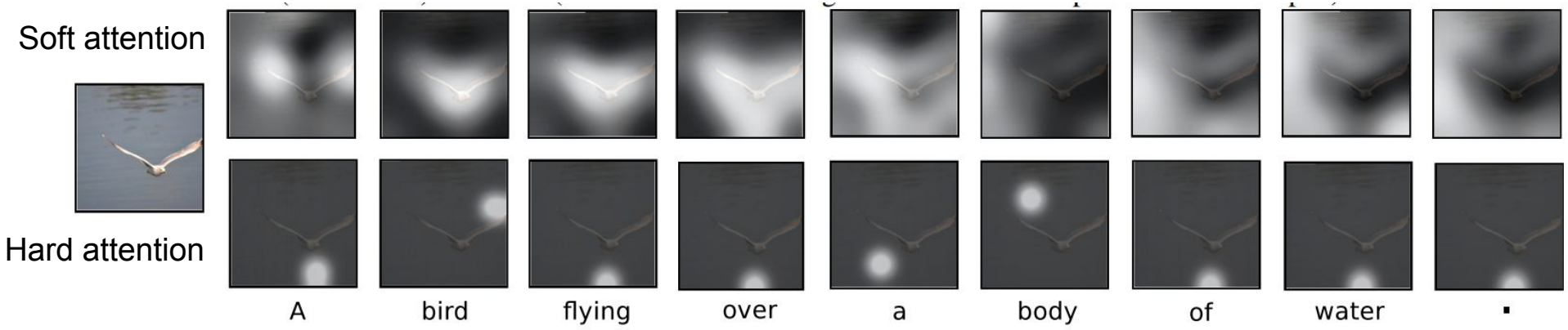
**Soft attention:**
Summarize ALL locations
$z = p_a a + p_b b + p_c c + p_d d$

Derivative dz/dp is nice!
Train with gradient descent

**Hard attention**:
Sample ONE location according to p, z = that vector

With argmax, dz/dp is zero almost everywhere …
Can't use gradient descent; need reinforcement learning

# Soft Attention for Captioning




Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

# Soft Attention for Captioning

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

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

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

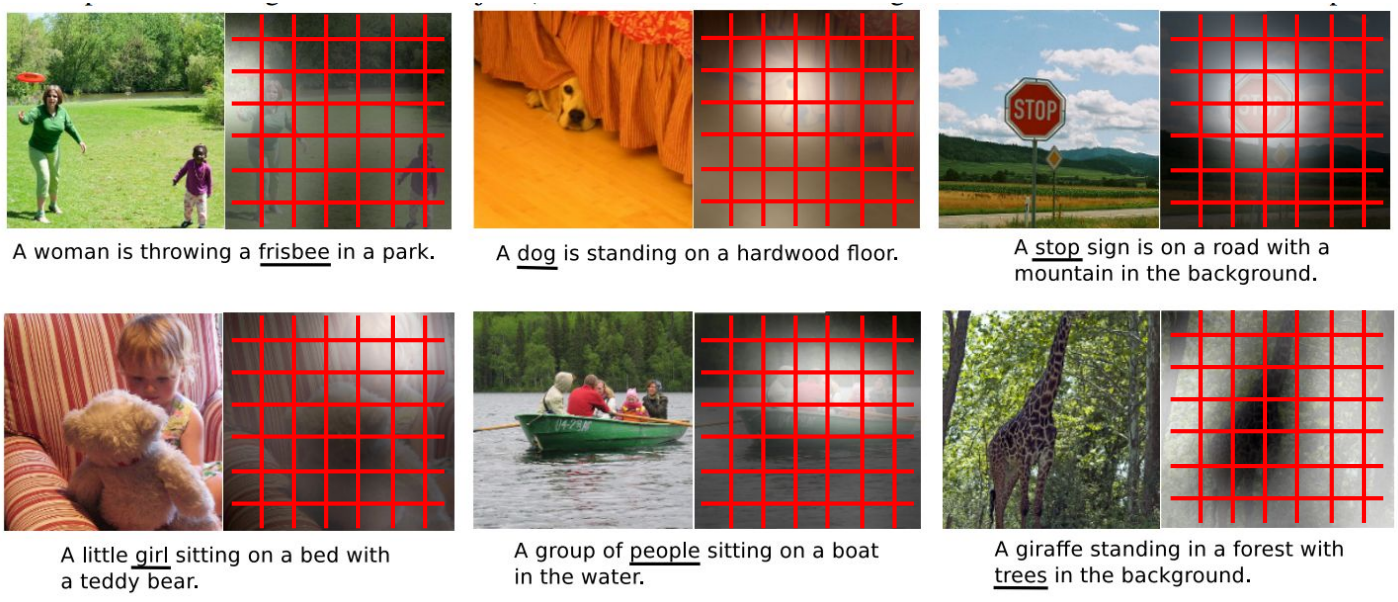
A group of people sitting on a boat in the water.

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

Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Soft Attention for Captioning

Attention constrained to fixed grid! We'll come back to this ….




Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

# Soft Attention for Everything!

**Machine Translation, attention over input:**
- Luong et al, "Effective Approaches to Attention-based Neural Machine Translation," EMNLP 2015

**Speech recognition, attention over input sounds:**
- Chan et al, "Listen, Attend, and Spell", arXiv 2015
- Chorowski et al, "Attention-based models for Speech Recognition", NIPS 2015

***+Local+Global*: Someone is frying a fish in a pot**

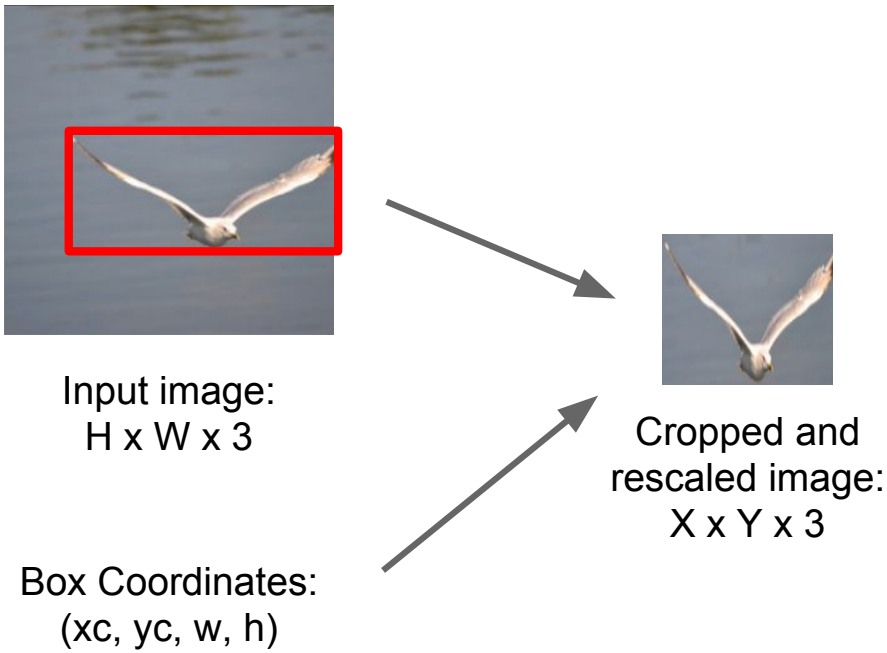
**Video captioning, attention over input frames:**
- Yao et al, "Describing Videos by Exploiting Temporal Structure", ICCV 2015


**Image, question to answer, attention over image:**
- Xu and Saenko, "Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for Visual Question Answering", arXiv 2015
- Zhu et al, "Visual7W: Grounded Question Answering in Images", arXiv 2015

Attending to Arbitrary Regions:

**Spatial Transformer Networks**

# Spatial Transformer Networks




Jaderberg et al, “Spatial Transformer Networks”, NIPS 2015

# Spatial Transformer Networks

Input image:
H x W x 3

Can we make this
function differentiable?

Cropped and
rescaled image:
X x Y x 3

Box Coordinates:
(xc, yc, w, h)

Jaderberg et al, “Spatial Transformer Networks”, NIPS 2015

# Spatial Transformer Networks

Input image:
H x W x 3

Can we make this function differentiable?

Cropped and rescaled image:
X x Y x 3

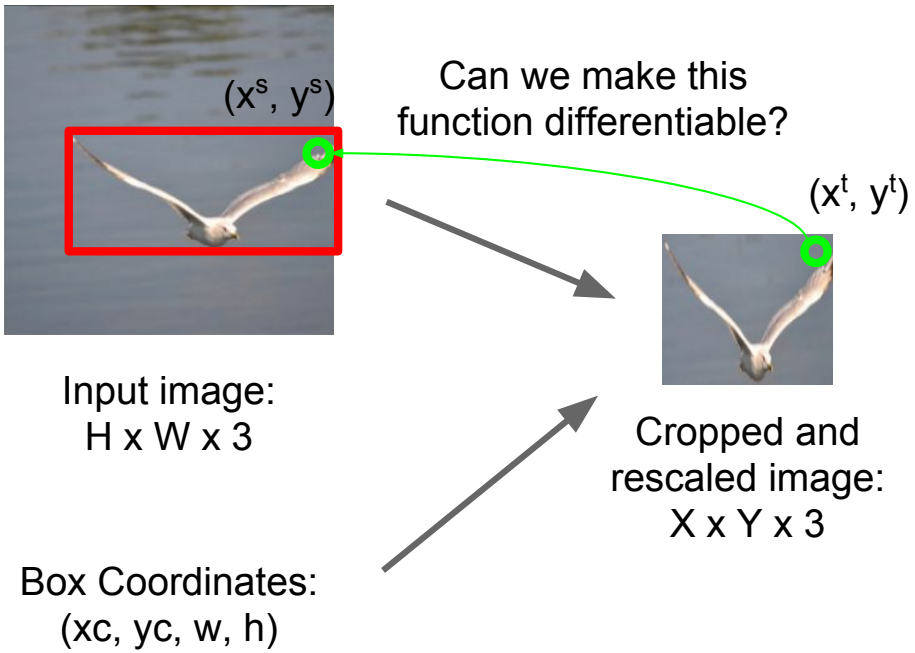
Box Coordinates:
(xc, yc, w, h)

**Idea**: Function mapping *pixel coordinates* (xt, yt) of output to *pixel coordinates* (xs, ys) of input

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

Jaderberg et al, “Spatial Transformer Networks”, NIPS 2015

# Spatial Transformer Networks

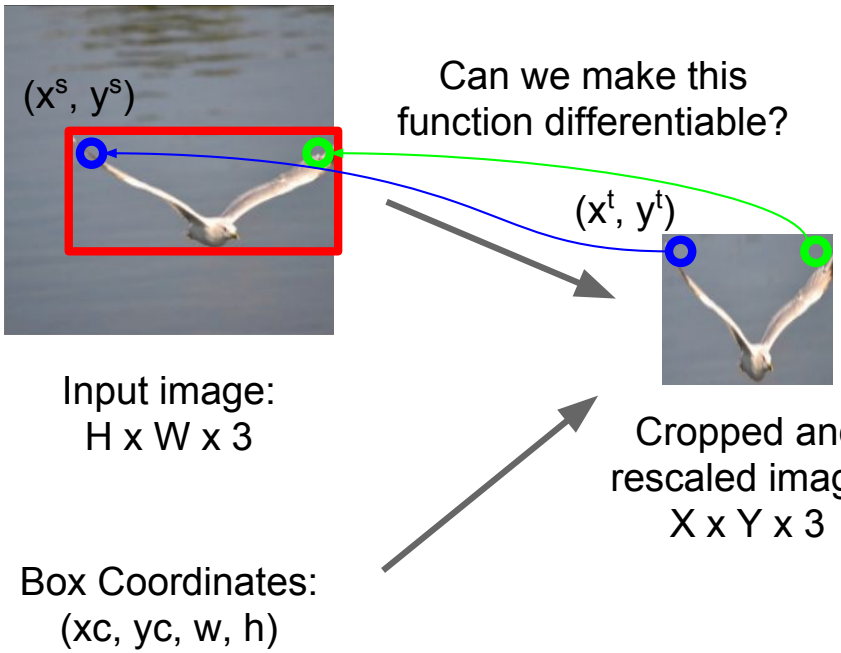



**Idea**: Function mapping *pixel coordinates* (xt, yt) of output to *pixel coordinates* (xs, ys) of input

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

Jaderberg et al, “Spatial Transformer Networks”, NIPS 2015

# Spatial Transformer Networks

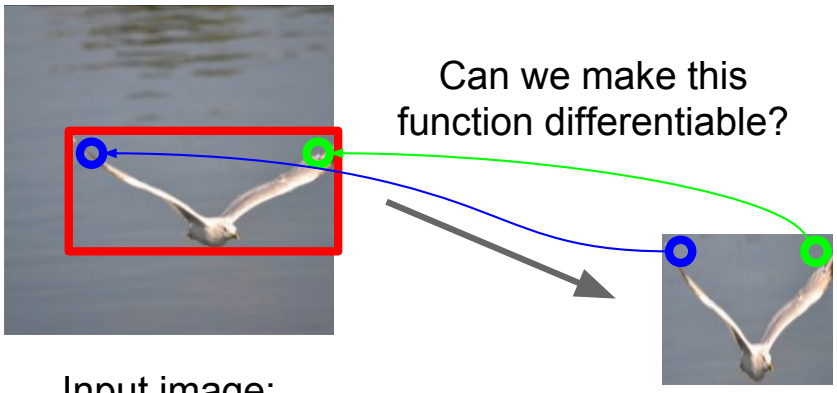



**Idea**: Function mapping *pixel coordinates* (xt, yt) of output to *pixel coordinates* (xs, ys) of input

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

Jaderberg et al, “Spatial Transformer Networks”, NIPS 2015

# Spatial Transformer Networks



Can we make this function differentiable?

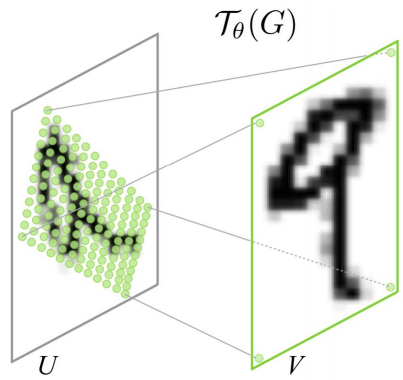
Input image:
H x W x 3

Box Coordinates:
(xc, yc, w, h)

Cropped and rescaled image:
X x Y x 3

**Idea**: Function mapping *pixel coordinates* (xt, yt) of output to *pixel coordinates* (xs, ys) of input

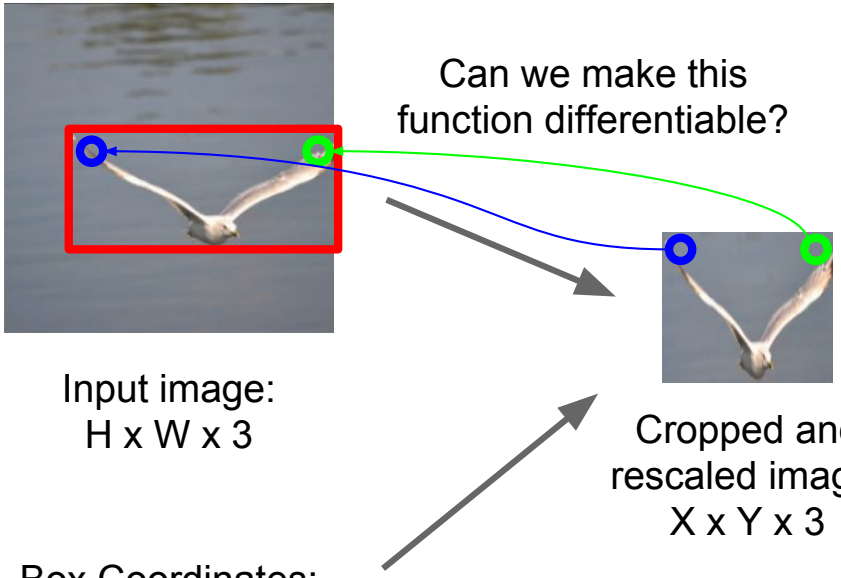
$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$




Repeat for all pixels in *output* to get a **sampling grid**

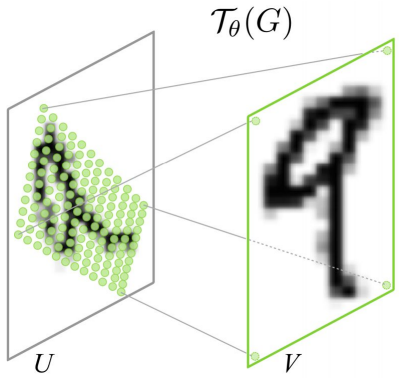
Jaderberg et al, "Spatial Transformer Networks", NIPS 2015

# Spatial Transformer Networks




**Idea**: Function mapping *pixel coordinates* (xt, yt) of output to *pixel coordinates* (xs, ys) of input

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

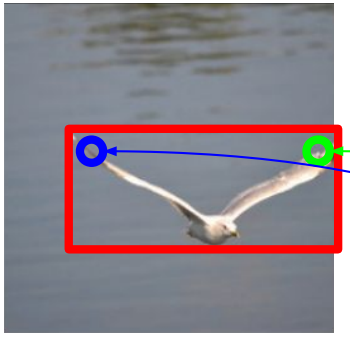



Repeat for all pixels in *output* to get a **sampling grid**

Then use **bilinear interpolation** to compute output

Jaderberg et al, "Spatial Transformer Networks", NIPS 2015

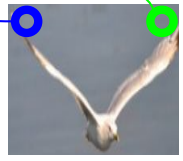
# Spatial Transformer Networks



Input image:
H x W x 3

Box Coordinates:
(xc, yc, w, h)

Can we make this function differentiable?

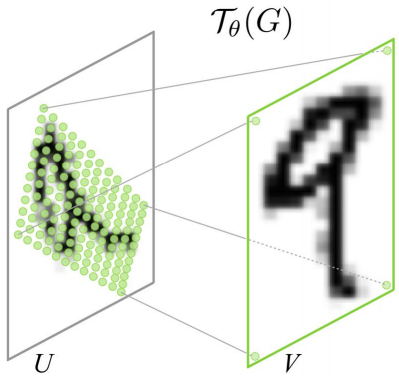


Cropped and rescaled image:
X x Y x 3

**Idea**: Function mapping *pixel coordinates* (xt, yt) of output to *pixel coordinates* (xs, ys) of input

Network attends to input by predicting $\theta$

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

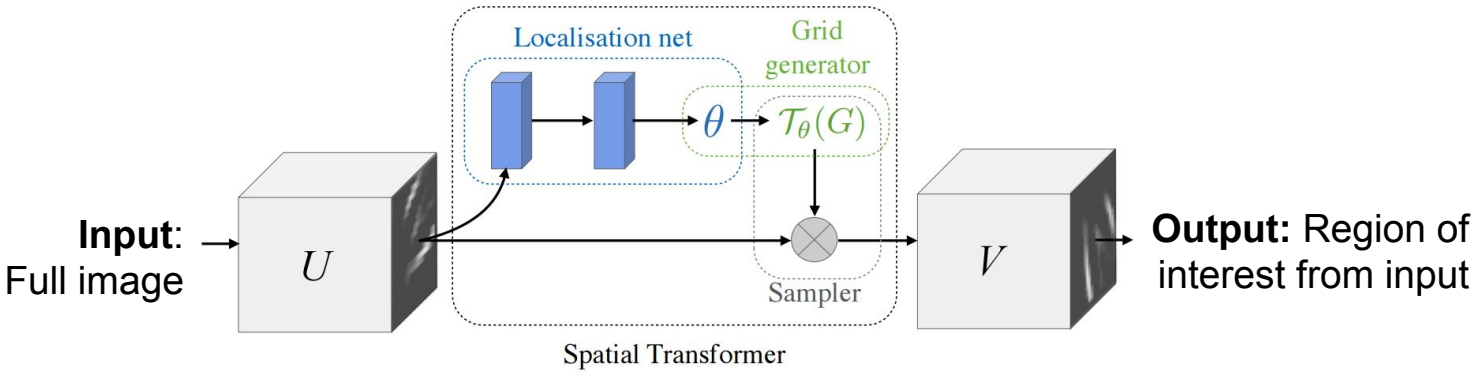



Repeat for all pixels in *output* to get a **sampling grid**

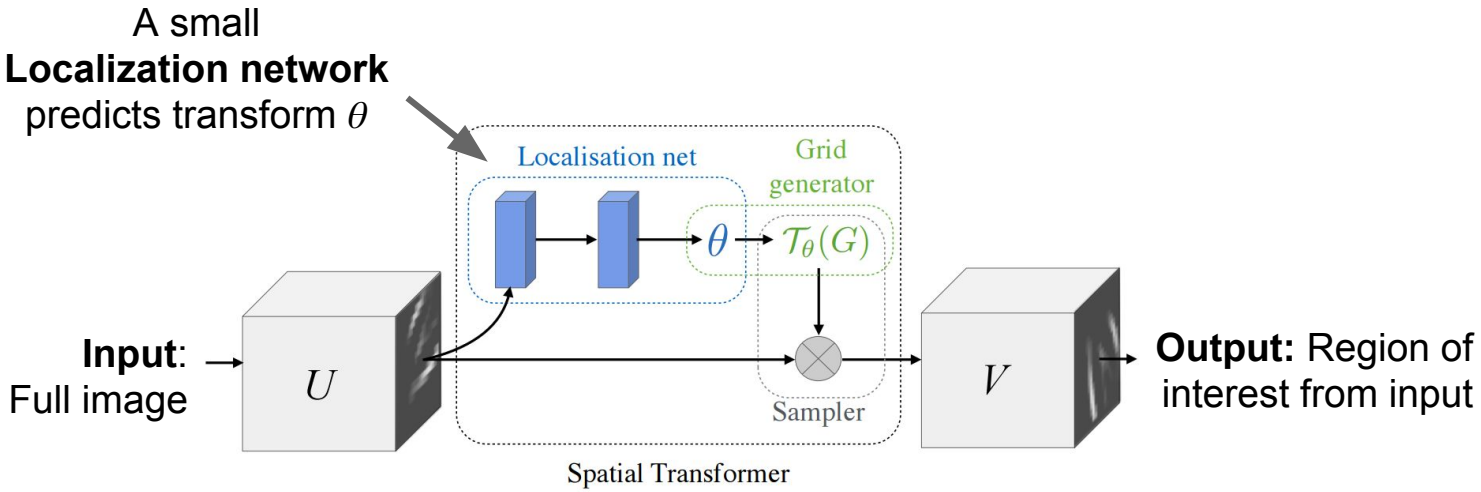
Then use **bilinear interpolation** to compute output

Jaderberg et al, "Spatial Transformer Networks", NIPS 2015

# Spatial Transformer Networks



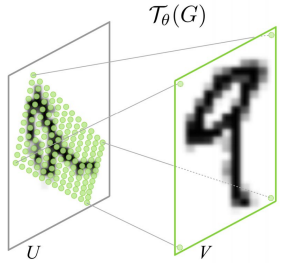
# Spatial Transformer Networks



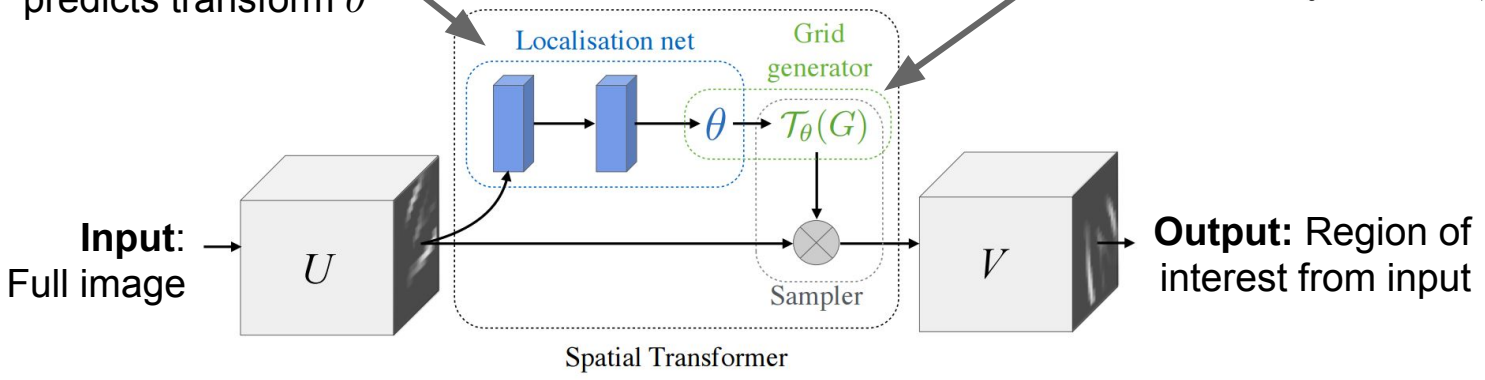
# Spatial Transformer Networks

**Grid generator** uses $\theta$ to compute sampling grid

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$




A small **Localization network** predicts transform $\theta$



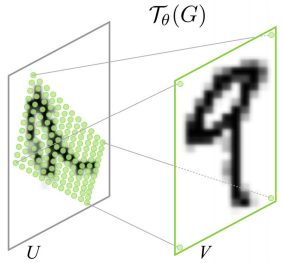

**Input**: Full image

**Output:** Region of interest from input

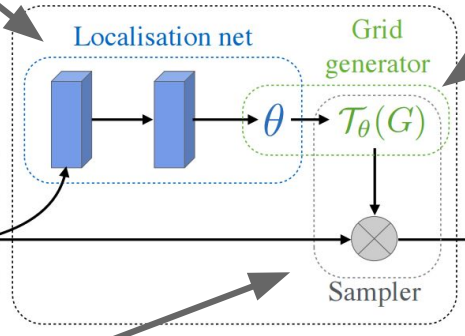
# Spatial Transformer Networks

**Grid generator** uses $\theta$ to compute sampling grid

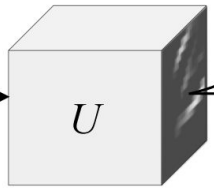
$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

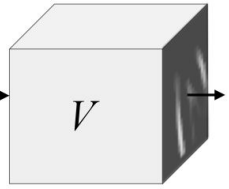



A small **Localization network** predicts transform $\theta$




**Input**: Full image




**Output:** Region of interest from input

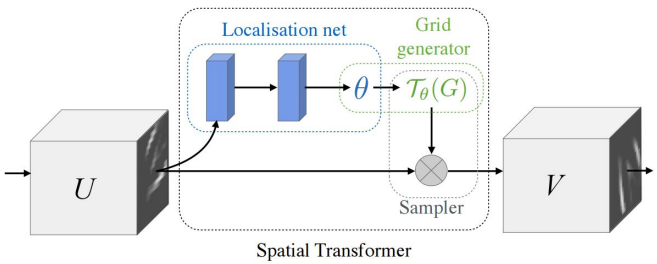
Spatial Transformer

**Sampler** uses bilinear interpolation to produce output

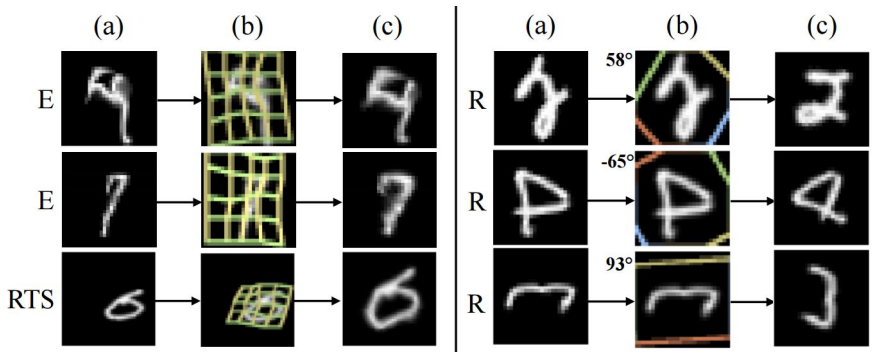
$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|)$$

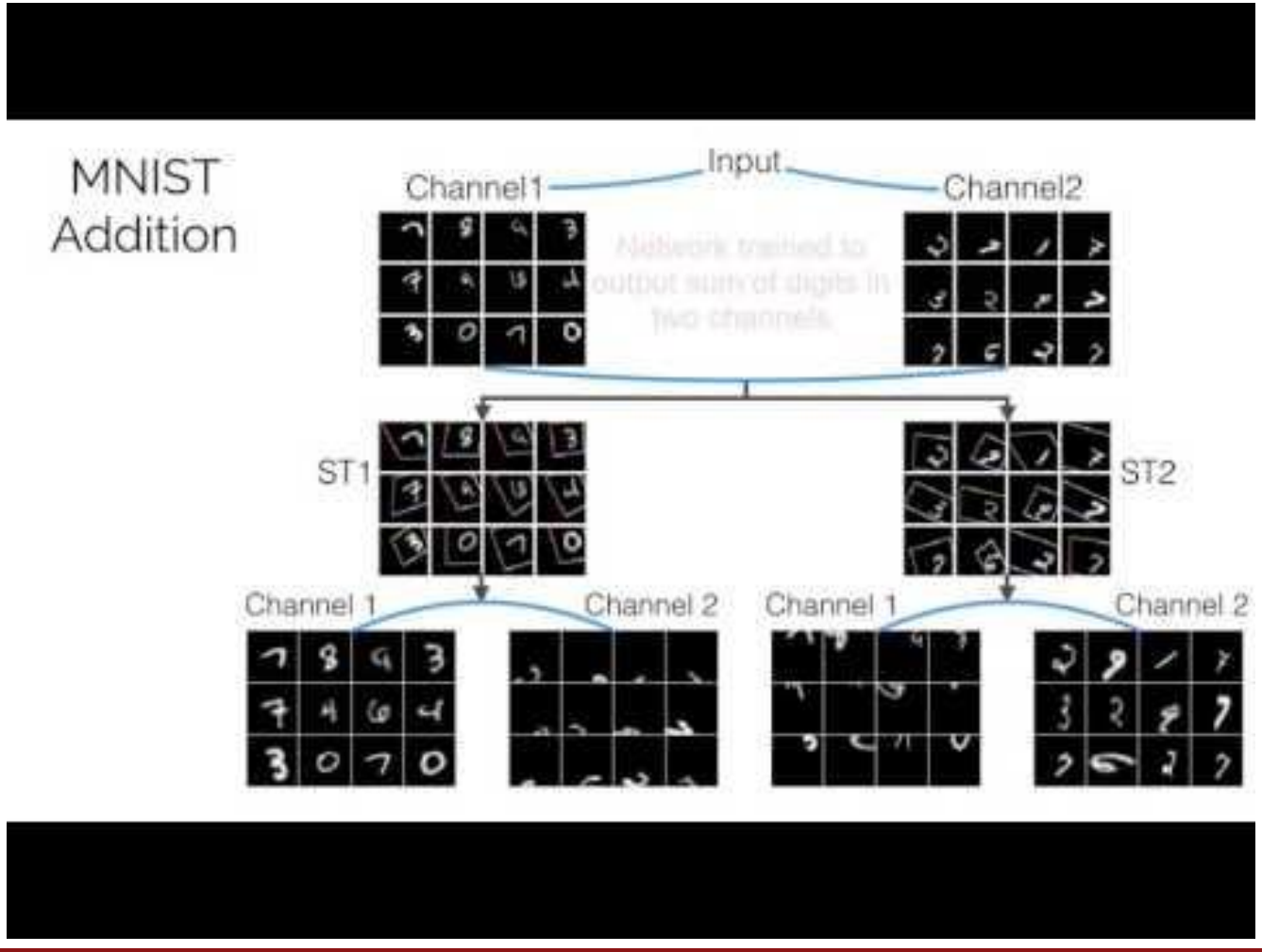
# Spatial Transformer Networks

Differentiable “attention / transformation” module




Insert spatial transformers into a classification network and it learns to attend and transform the input




MNIST
Addition
Input
Channel1
Channel2
ST1
ST2
Channel 1
Channel 2
Channel 1
Channel 2