Lecture 13:

Segmentation and Attention
Administrative

- Assignment 3 due tonight!
- We are reading your milestones
Last time: Software Packages

Caffe
No need to write code!
1. Convert data (run a script)
2. Define net (edit prototxt)
3. Define solver (edit prototxt)
4. Train (with pretrained weights)

Torch

Theano

Lasagne

TensorFlow

Keras

Fei-Fei Li & Andrej Karpathy & Justin Johnson
Today

- **Segmentation**
  - Semantic Segmentation
  - Instance Segmentation

- **(Soft) Attention**
  - Discrete locations
  - Continuous locations (Spatial Transformers)
But first....

ImageNet Classification Error (Top 5)

- 2012 (AlexNet): 16.4
- 2013 (ZF): 11.7
- 2014 (VGG): 7.3
- 2014 (GoogLeNet): 6.7
- 2015 (ResNet): 3.57
But first….

New ImageNet Record today!

Inception-v4

Inception-v4

Inception-v4


Fei-Fei Li & Andrej Karpathy & Justin Johnson
Inception-v4

Inception-ResNet-v2

9 layers
Inception-ResNet-v2

3 layers
5 x 4 layers
9 layers

Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 13 - 12  24 Feb 2016
Inception-ResNet-v2

- **Softmax**
  - Output: 1000
- **Dropout (keep 0.8)**
  - Output: 1702
- **Average Pooling**
  - Output: 1752
  - 5 x Inception-resnet-C
    - Output: 8x8x1792
    - **Reduction-B**
      - Output: 17x17x64
      - 10 x Inception-resnet-B
        - Output: 17x17x64
        - **Reduction-A**
          - Output: 17x17x64
          - 5 x Inception-resnet-A
            - Output: 35x35x256
            - **Stern**
              - Output: 35x35x256
              - Input (299x299x3)

- **Filter concat**
  - 3x3 MaxPool (stride 2)
  - 3x3 Conv (256 stride 2)
  - 1x1 Conv (128)
  - Previous Layer

- **Relu activation**
  - 1x1 Conv (128)
  - 1x1 Conv (192)
  - 7x1 Conv (192)
  - 1x7 Conv (192)
  - 1x1 Conv (160)

- **Whitening**
  - Output: 2048
Inception-ResNet-v2

- Softmax
- Dropout (keep 0.8)
- Average Pooling

5 x 4 layers
- Reduction-B
- 10 x Inception-resnet-B
- Reduction-A
- 5 x Inception-resnet-A
- Stem

5 x 3 layers
- 9 layers

10 x 4 layers
- 3 layers

75 layers

x 5

Relu activation

1x1 Conv (2048 Linear)
- 3x1 Conv (256)
- 1x3 Conv (224)
- 1x1 Conv (192)

Relu activation
Inception-ResNet-v2

Residual and non-residual converge to similar value, but residual learns faster
Today

● Segmentation
  ○ Semantic Segmentation
  ○ Instance Segmentation

● (Soft) Attention
  ○ Discrete locations
  ○ Continuous locations (Spatial Transformers)
Computer Vision Tasks

**Classification**

- CAT

**Classification + Localization**

- CAT

**Object Detection**

- CAT, DOG, DUCK

**Segmentation**

- CAT, DOG, DUCK

Single object

Multiple objects
Computer Vision Tasks

Classification + Localization

Object Detection

Lecture 8

Fei-Fei Li & Andrej Karpathy & Justin Johnson
Lecture 13 1919 24 Feb 2016
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

Extract patch

Run through a CNN

CNN
Semantic Segmentation

Extract patch → Run through a CNN → Classify center pixel

CNN

COW
Semantic Segmentation

Extract patch

Run through a CNN

Classify center pixel

Repeat for every pixel

COW

cow
grass
Semantic Segmentation

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

Smaller output due to pooling
Semantic Segmentation: Multi-Scale

Semantic Segmentation: Multi-Scale

Resize image to multiple scales

Semantic Segmentation: Multi-Scale

Semantic Segmentation: Multi-Scale

Semantic Segmentation: Multi-Scale

- Resize image to multiple scales
- Run one CNN per scale
- Upscale outputs and concatenate

External “bottom-up” segmentation

---

Semantic Segmentation: Multi-Scale

- Resize image to multiple scales
- Run one CNN per scale
- Upscale outputs and concatenate
- Combine everything for final outputs

External "bottom-up" segmentation

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

Apply CNN once to get labels

Apply AGAIN to refine labels

And again!

Same CNN weights: recurrent convolutional network

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

Semantic Segmentation: Upsampling

Learnable upsampling!

Semantic Segmentation: Upsampling

Semantic Segmentation: Upsampling

Semantic Segmentation: Upsampling

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

Input: 4 x 4

Dot product between filter and input

Output: 4 x 4
Learnable Upsampling: “Deconvolution”

Typical 3 x 3 convolution, stride 1 pad 1

- Input: 4 x 4
- Dot product between filter and input
- Output: 4 x 4
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

Input: 4 x 4

Dot product between filter and input

Output: 2 x 2
Learnable Upsampling: “Deconvolution”

Typical 3 x 3 convolution, stride 2 pad 1

Dot product between filter and input

Input: 4 x 4

Output: 2 x 2
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

Input: 2 x 2

Input gives weight for filter

Output: 4 x 4
Learnable Upsampling: “Deconvolution”

3 x 3 “deconvolution”, stride 2 pad 1

Input gives weight for filter

Input: 2 x 2
Output: 4 x 4
Learnable Upsampling: “Deconvolution”

3 x 3 “deconvolution”, stride 2 pad 1

Input gives weight for filter

Sum where output overlaps

Input: 2 x 2

Output: 4 x 4
Learnable Upsampling: “Deconvolution”

- 3 x 3 “deconvolution”, stride 2 pad 1
- Input gives weight for filter
- Sum where output overlaps
- Same as backward pass for normal convolution!
Learnable Upsampling: “Deconvolution”

3 x 3 “deconvolution”, stride 2 pad 1

Input gives weight for filter

Sum where output overlaps

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”

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


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

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

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

Learnable Upsampling: “Deconvolution”

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


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


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

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

Great explanation in appendix
Semantic Segmentation: Upsampling

Semantic Segmentation: Upsampling


6 days of training on Titan X…
Instance Segmentation
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
Instance Segmentation

Similar to R-CNN, but with segments

Instance Segmentation

Similar to R-CNN, but with segments

Hariharan et al, "Simultaneous Detection and Segmentation", ECCV 2014
Instance Segmentation

Similar to R-CNN, but with segments

Instance Segmentation

Similar to R-CNN, but with segments

Instance Segmentation

Similar to R-CNN, but with segments

Proposal Generation → External Segment proposals → Feature Extraction → Region Classification

Mask out background with mean image

Instance Segmentation

Similar to R-CNN, but with segments

Proposal Generation
External Segment proposals
Feature Extraction
Region Classification
Region Refinement

Mask out background with mean image

Instance Segmentation: Hypercolumns

Instance Segmentation: Hypercolumns

Instance Segmentation: Cascades

Similar to Faster R-CNN

Won COCO 2015 challenge (with ResNet)

Instance Segmentation: Cascades

Similar to Faster R-CNN

Won COCO 2015 challenge (with ResNet)

Instance Segmentation: Cascades

Similar to Faster R-CNN

Won COCO 2015 challenge (with ResNet)

Instance Segmentation: Cascades


Won COCO 2015 challenge (with ResNet)
Instance Segmentation: Cascades

Similar to Faster R-CNN

Region proposal network (RPN)

- box instances (RoIs)
- Reshape boxes to fixed size, figure / ground
- logistic regression
- Mask out background, predict object class

Won COCO 2015 challenge (with ResNet)

Instance Segmentation: Cascades

Similar to Faster R-CNN

Region proposal network (RPN)

- box instances (RoIs)
- Reshape boxes to fixed size, figure / ground
- logistic regression
- mask instances

Mask out background, predict object class

Learn entire model end-to-end!

Won COCO 2015 challenge (with ResNet)

Instance Segmentation: Cascades

Segmentation Overview

- Semantic segmentation
  - Classify all pixels
  - Fully convolutional models, downsample then upsample
  - Learnable upsampling: fractionally strided convolution
  - Skip connections can help

- Instance Segmentation
  - Detect instance, generate mask
  - Similar pipelines to object detection
Attention Models
Recall: RNN for Captioning

Image: H x W x 3
Recall: RNN for Captioning

Image: $H \times W \times 3$

Features: $D$
Recall: RNN for Captioning

- Image: \(H \times W \times 3\)
- Features: \(D\)
- Hidden state: \(H\)
Recall: RNN for Captioning

Image:
H x W x 3

Features:
D

Hidden state:
H

Distribution over vocab

First word

y1

h1

h0

d1
Recall: RNN for Captioning

Image: $H \times W \times 3$

Features: $D$

Hidden state: $H$

Distribution over vocab

- $d_1$
- $d_2$

First word

- $y_1$

Second word

- $y_2$
Recall: RNN for Captioning

Image: $H \times W \times 3$

Features: $D$

Hidden state: $H$

First word

Second word

Distribution over vocab

RNN only looks at whole image, once
Recall: RNN for Captioning

Image:

Features: $D$

Hidden state: $H$

$y_1$: First word

$y_2$: Second word

Distribution over vocab

What if the RNN looks at different parts of the image at each timestep?

RNN only looks at whole image, once
Soft Attention for Captioning

Image: H x W x 3

Features: L x D

Soft Attention for Captioning

Image: $H \times W \times 3$

Features: $L \times D$

$h_0$

Soft Attention for Captioning

Image: 
H x W x 3

Features: 
L x D

Distribution over 
L locations

CNN

a1

h0

Soft Attention for Captioning

Image: \( H \times W \times 3 \)

Features: \( L \times D \)

Weighted combination of features

Weighted features: \( z_1 \)

Distribution over \( L \) locations

Soft Attention for Captioning

Image: H x W x 3

Features: L x D

Distribution over L locations

h0

Weighted combination of features

Weighted features: D

z1

y1

First word

a1

Soft Attention for Captioning

- CNN
- Image: $H \times W \times 3$
- Features: $L \times D$
- Distribution over $L$ locations
- Distribution over vocab
- $a_1$, $a_2$, $d_1$
- $h_0$, $h_1$
- Weighted combination of features
- Weighted features: $D$
- $z_1$, $y_1$
- First word

Soft Attention for Captioning

Image: \(H \times W \times 3\)

Features: \(L \times D\)

Weighted combination of features

Weighted features: \(D\)

First word

Distribution over \(L\) locations

Distribution over vocab

Soft Attention for Captioning

Soft Attention for Captioning

Image: $H \times W \times 3$

Features: $L \times D$

Weighted combination of features

Weighted features: $D$

First word

Distribution over L locations

Distribution over vocab

Soft Attention for Captioning

Guess which framework was used to implement?

Soft Attention for Captioning

Guess which framework was used to implement?

Crazy RNN = Theano

**Soft vs Hard Attention**

Image: H x W x 3

Grid of features (Each D-dimensional)

- a
- b
- c
- d

Distribution over grid locations:

\[ p_a + p_b + p_c + p_d = 1 \]

From RNN:

Soft vs Hard Attention

Image: 
H x W x 3

Grid of features 
(Each D-dimensional)

\[
\begin{array}{cc}
a & b \\
c & d \\
\end{array}
\]

From RNN:

Distribution over 
grid locations
\[p_a + p_b + p_c + p_d = 1\]

Context vector \(z\) 
(D-dimensional)

\[Xu \ et \ al, \ "Show, \ Attend \ and \ Tell: \ Neural \ Image \ Caption \ Generation \ with \ Visual \ Attention", \ ICML \ 2015\]
Soft vs Hard Attention

Image: \( H \times W \times 3 \)

Grid of features (Each D-dimensional)

\[
\begin{array}{cc}
a & b \\
c & d \\
\end{array}
\]

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

Context vector \( z \) (D-dimensional)

From RNN:

Distribution over grid locations
\[ p_a + p_b + p_c + p_d = 1 \]

### Soft vs Hard Attention

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

Derivative \( \frac{dz}{dp} \) is nice!
Train with gradient descent

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

With argmax, \( \frac{dz}{dp} \) is zero almost everywhere ...
Can’t use gradient descent; need reinforcement learning

---


---

Image: \( H \times W \times 3 \)

Grid of features (Each D-dimensional)

**From RNN:**

Distribution over grid locations
\[ p_a + p_b + p_c + p_d = 1 \]

---

Context vector \( z \) (D-dimensional)
Soft Attention for Captioning

Soft Attention for Captioning

Soft Attention for Captioning

Soft Attention for Translation

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

Soft Attention for Translation

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

Soft Attention for Translation

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

Soft Attention for Translation

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

Distribution over input words

Soft Attention for Everything!

Machine Translation, attention over input:

Speech recognition, attention over input sounds:

Video captioning, attention over input frames:

Image, question to answer, attention over image:
Attending to arbitrary regions?

Attention mechanism from Show, Attend, and Tell only lets us softly attend to fixed grid positions … can we do better?
Attending to Arbitrary Regions

- Read text, generate handwriting using an RNN
- Attend to arbitrary regions of the output by predicting params of a mixture model

Graves, “Generating Sequences with Recurrent Neural Networks”, arXiv 2013
Attending to Arbitrary Regions

- Read text, generate handwriting using an RNN
- Attend to arbitrary regions of the output by predicting params of a mixture model

Graves, “Generating Sequences with Recurrent Neural Networks”, arXiv 2013
Attending to Arbitrary Regions

- Read text, generate handwriting using an RNN
- Attend to arbitrary regions of the output by predicting params of a mixture model

Which are real and which are generated?

Graves, “Generating Sequences with Recurrent Neural Networks”, arXiv 2013
Attending to Arbitrary Regions: DRAW

Classify images by attending to arbitrary regions of the input

Attending to Arbitrary Regions: DRAW

**Classify** images by attending to arbitrary regions of the *input*

**Generate** images by attending to arbitrary regions of the *output*

Gregor et al, "DRAW: A Recurrent Neural Network For Image Generation", ICML 2015
Attending to Arbitrary Regions: Spatial Transformer Networks

Attention mechanism similar to DRAW, but easier to explain

Spatial Transformer Networks

Input image: $H \times W \times 3$

Box Coordinates: $(x_c, y_c, w, h)$

Cropped and rescaled image: $X \times Y \times 3$

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


Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 13 - 120  24 Feb 2016
Spatial Transformer Networks

Can we make this function differentiable?

Input image: \( H \times W \times 3 \)

Box Coordinates: \((x_c, y_c, w, h)\)

Cropped and rescaled image: \( X \times Y \times 3 \)


Idea: Function mapping pixel coordinates \((x_t, y_t)\) of output to pixel coordinates \((x_s, y_s)\) of input

\[
\begin{pmatrix}
  x_s^i \\
  y_s^i
\end{pmatrix} =
\begin{bmatrix}
  \theta_{11} & \theta_{12} & \theta_{13} \\
  \theta_{21} & \theta_{22} & \theta_{23}
\end{bmatrix}
\begin{pmatrix}
  x_t^i \\
  y_t^i \\
  1
\end{pmatrix}
\]

Spatial Transformer Networks

Input image: $H \times W \times 3$

Box Coordinates: $(x_c, y_c, w, h)$

Cropped and rescaled image: $X \times Y \times 3$

Can we make this function differentiable?

Idea: Function mapping pixel coordinates $(x_t, y_t)$ of output to pixel coordinates $(x_s, y_s)$ of input

$$
\begin{pmatrix}
    x^s_i \\
    y^s_i
\end{pmatrix} =
\begin{bmatrix}
    \theta_{11} & \theta_{12} & \theta_{13} \\
    \theta_{21} & \theta_{22} & \theta_{23}
\end{bmatrix}
\begin{pmatrix}
    x^t_i \\
    y^t_i \\
    1
\end{pmatrix}
$$

Spatial Transformer Networks

Can we make this function differentiable?

**Idea:** Function mapping *pixel coordinates* \((x_t, y_t)\) of output to *pixel coordinates* \((x_s, y_s)\) of input

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

Input image: \(H \times W \times 3\)

Box Coordinates: \((x_c, y_c, w, h)\)

Cropped and rescaled image: \(X \times Y \times 3\)

Spatial Transformer Networks

**Input image:** $H \times W \times 3$

**Box Coordinates:** $(x_c, y_c, w, h)$

**Cropped and rescaled image:** $X \times Y \times 3$

Can we make this function differentiable?

**Idea:** Function mapping pixel coordinates $(x_t, y_t)$ of output to pixel coordinates $(x_s, y_s)$ of input

\[
\begin{pmatrix}
    x^s_i \\
    y^s_i
\end{pmatrix} =
\begin{bmatrix}
    \theta_{11} & \theta_{12} & \theta_{13} \\
    \theta_{21} & \theta_{22} & \theta_{23}
\end{bmatrix}
\begin{pmatrix}
    x^t_i \\
    y^t_i \\
    1
\end{pmatrix}
\]

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

Spatial Transformer Networks

Input image: \( H \times W \times 3 \)

Box Coordinates: \( (x_c, y_c, w, h) \)

Can we make this function differentiable?

Cropped and rescaled image: \( X \times Y \times 3 \)

Idea: Function mapping pixel coordinates \((x_t, y_t)\) of output to pixel coordinates \((x_s, y_s)\) of input

\[
\begin{pmatrix}
  x_s^i \\
  y_s^i
\end{pmatrix} =
\begin{bmatrix}
  \theta_{11} & \theta_{12} & \theta_{13} \\
  \theta_{21} & \theta_{22} & \theta_{23}
\end{bmatrix}
\begin{pmatrix}
  x_t^i \\
  y_t^i \\
  1
\end{pmatrix}
\]

Repeat for all pixels in output to get a sampling grid

Then use bilinear interpolation to compute output

Spatial Transformer Networks

Input image: $H \times W \times 3$

Box Coordinates: $(x_c, y_c, w, h)$

Cropped and rescaled image: $X \times Y \times 3$

Can we make this function differentiable?

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

Input: Full image

Output: Region of interest from input

$U \rightarrow \text{Spatial Transformer} \rightarrow V$
Spatial Transformer Networks

A small Localization network predicts transform $\theta$

Input: Full image

Output: Region of interest from input
Spatial Transformer Networks

A small Localization network predicts transform $\theta$

Grid generator uses $\theta$ to compute sampling grid

$$
\begin{pmatrix}
  x_i^s \\
  y_i^s \\
1
\end{pmatrix} =
\begin{bmatrix}
  \theta_{11} & \theta_{12} & \theta_{13} \\
  \theta_{21} & \theta_{22} & \theta_{23} \\
 0 & 0 & 1
\end{bmatrix}
\begin{pmatrix}
  x_i^t \\
  y_i^t \\
1
\end{pmatrix}
$$

Input: Full image

Output: Region of interest from input
Spatial Transformer Networks

A small Localization network predicts transform $\theta$

Input: Full image

Sampler uses bilinear interpolation to produce output

Output: Region of interest from input

Grid generator uses $\theta$ to compute sampling grid

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

Network trained to output sum of digits in two channels.
Attention Recap

- **Soft attention:**
  - Easy to implement: produce distribution over input locations, reweight features and feed as input
  - Attend to arbitrary input locations using spatial transformer networks

- **Hard attention:**
  - Attend to a single input location
  - Can’t use gradient descent!
  - Need **reinforcement learning!**